Robo Recall Modding

Starting Out

Creating Your First Robo Recall Mod

Getting up and running with your first Robo Recall mod

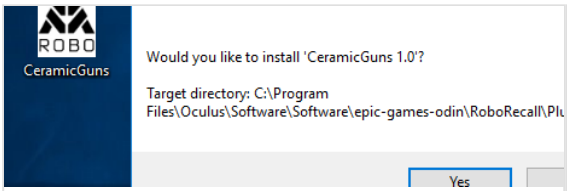
Guides

Enabling a Mod

How to make the mods you have installed for Robo Recall active in the game

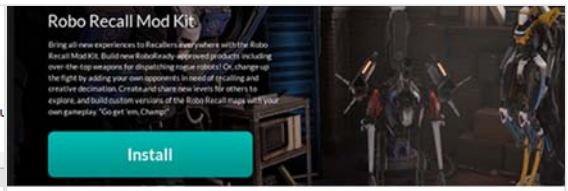
Overriding and Extending Functionality

Guide that walks you through overriding the existing functionality of an item from the game



Installing a Robo Recall Mod

Instructions for installing Robo Recall mods for use in the game



Installing the Robo Recall Mod Kit

Guide to downloading and installing the Robo Recall mod Kit used to make mods for the game

Adding Gameplay To Your Map

Guide that demonstrates adding gameplay elements to a custom map to work with the Robo Recall game

Troubleshooting Robo Recall Mods

Disagnosing and solving issues encountered when modding Robo Recall

Reference

Base Gun Reference

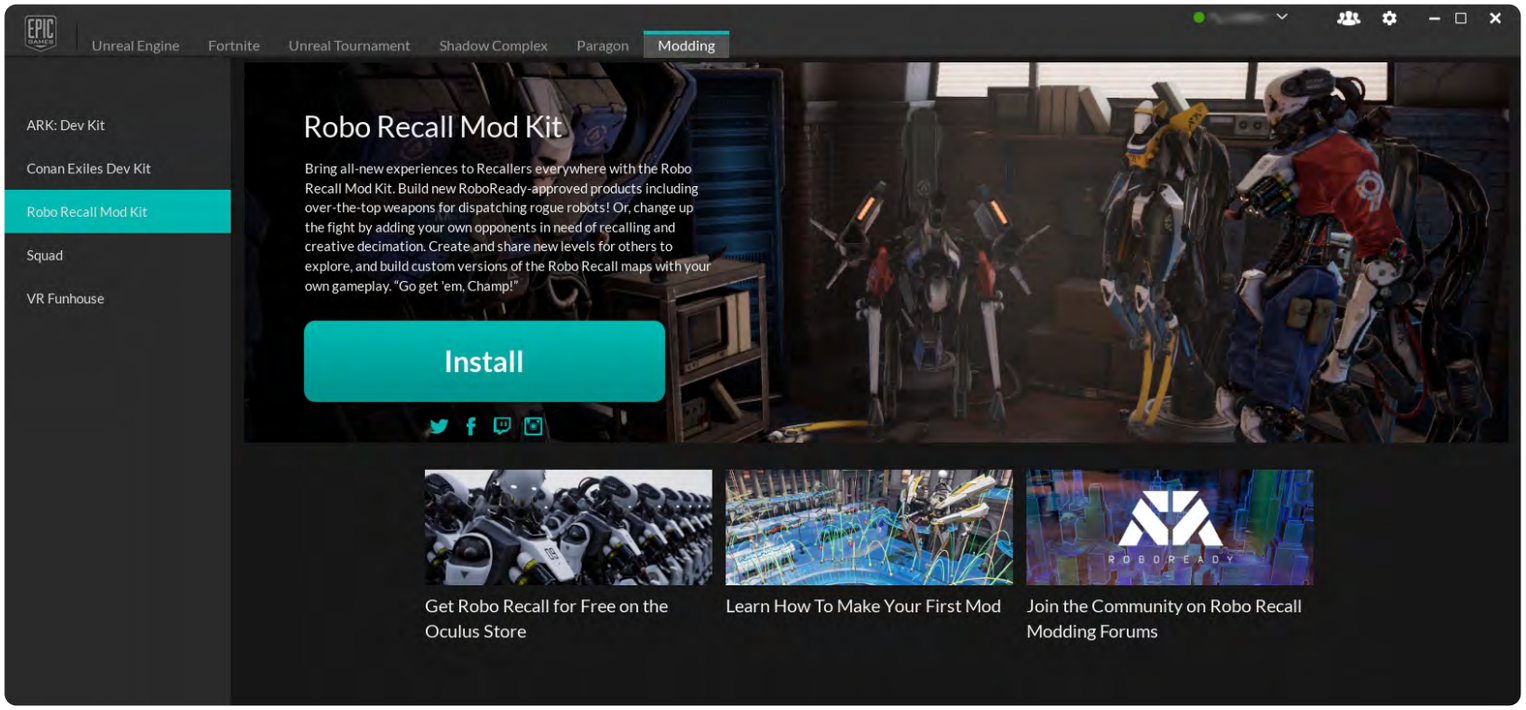
Creating Your First Robo Recall Mod

Steps

Click to Start ›

1. Install the Robo Recall Mod Kit

1. Sign up for an Epic ID
2. Download and install the Epic Games Launcher
3. Click **Install** to download the Robo Recall Mod Kit from the Epic Games Launcher.

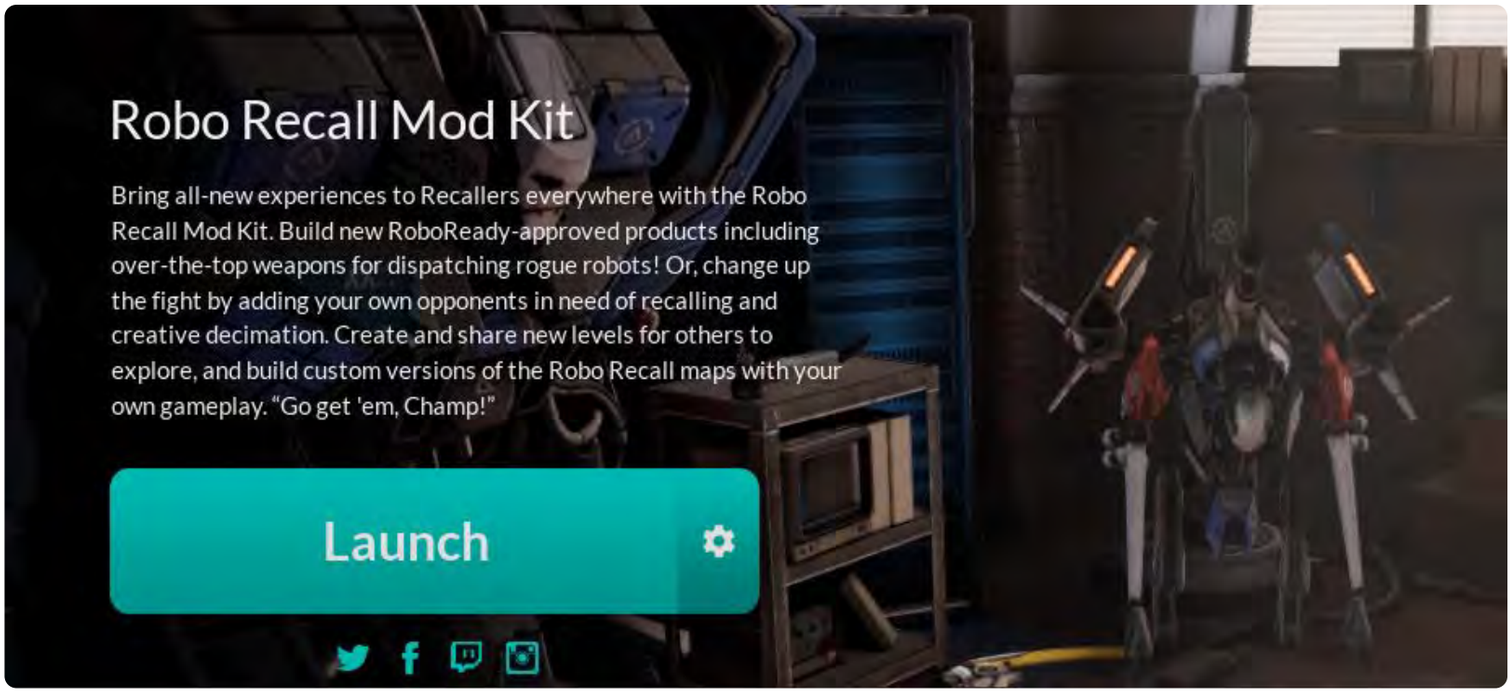


This can take some time as the Robo Recall Editor is quite large (20+ GB). Take this time to review some of the other Robo Recall Documentation or perhaps some of the Unreal Engine 4 getting started guides:

- **Level Designer Quick Start**
- **Blueprints Visual Scripting**
- **Programming Quick Start**
- **Managing Content**

Once the download is complete, head to the next step.

4. Launch the Robo Recall Editor.



Result

With the Robo Recall Editor open the editor will be waiting for you to choose a type of mod to create!

Previous Step | Creating Your First Robo Recall Mod | Next Step

2. Create a New Mod

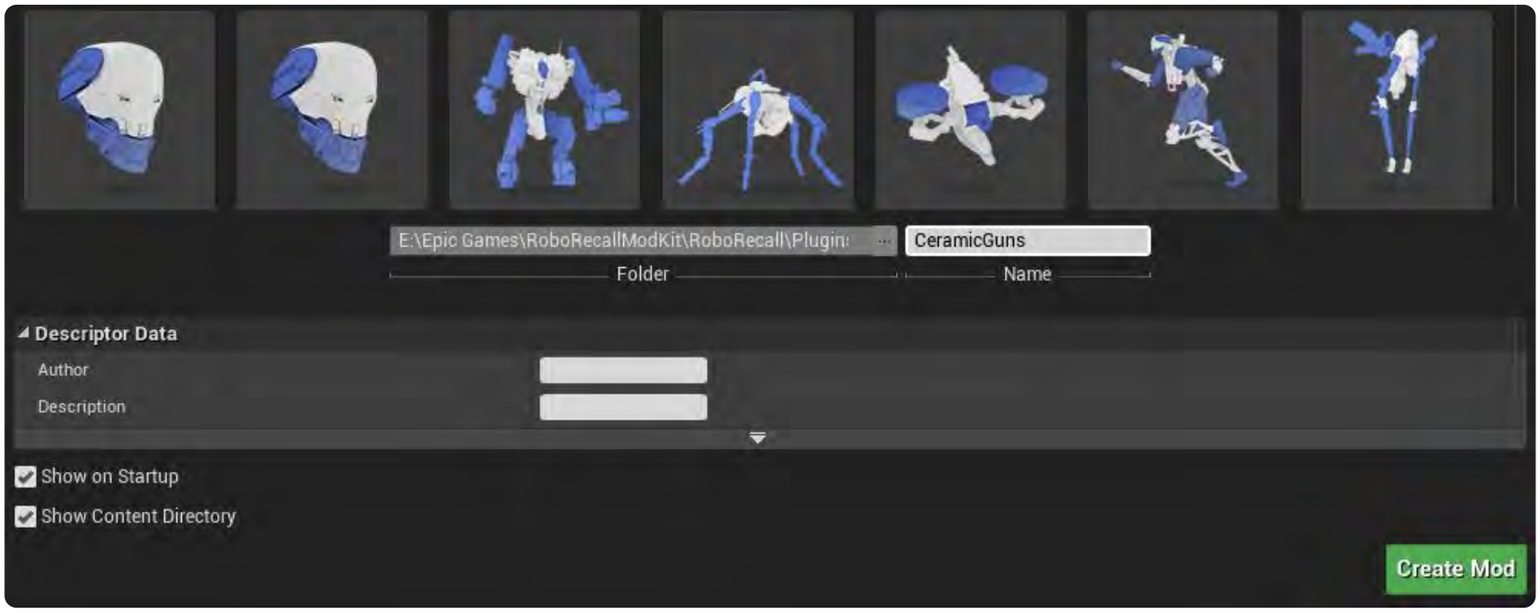
In this step, you'll be creating a new mod. While some of this is open ended (Do you want to make a new Robot? A new Gun? Level?), this is the first thing that needs to be done before you start modding Robo Recall.

Steps

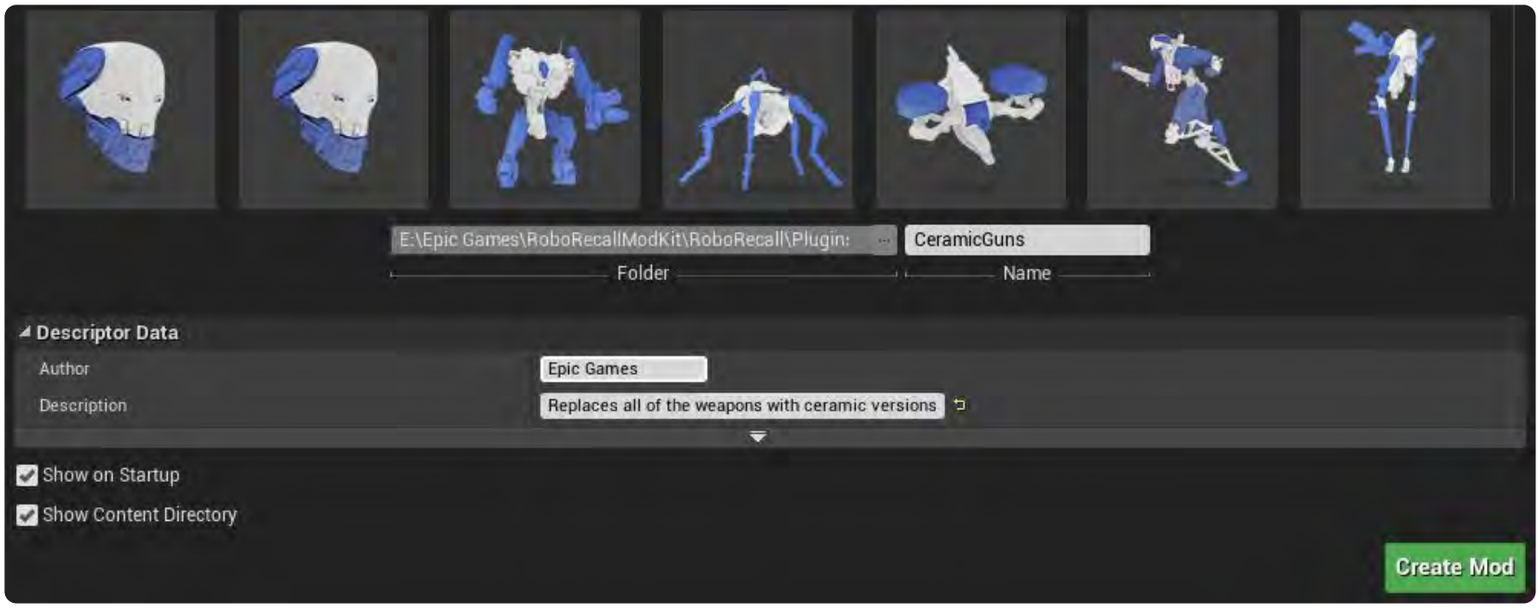
1. Click the **Create Mod** button on the Toolbar.
2. Select the type of mod you wish to create.

> You can multiselect in this dialog to create multiple assets at the same time. Feel free to add any other weapons, bots, etc… by **Ctrl Clicking** the assets you want to create in your mod.

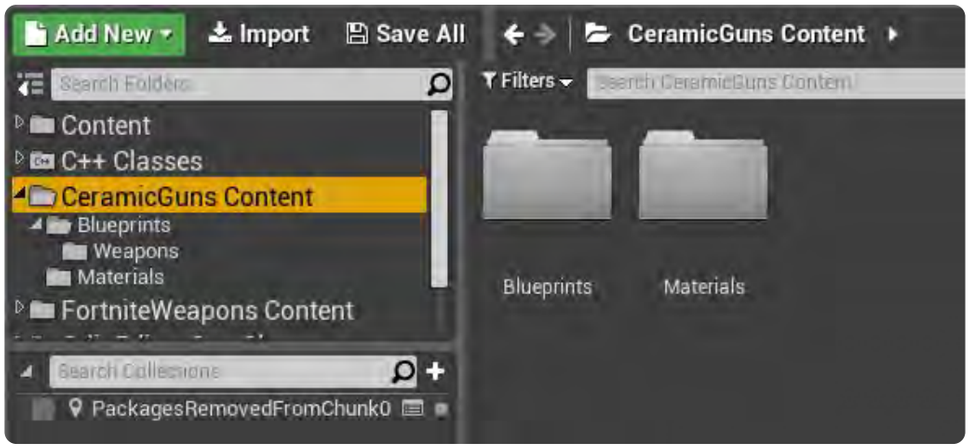
3. Enter a new mod name, in this case, we've chosen **GrenadeGun**.



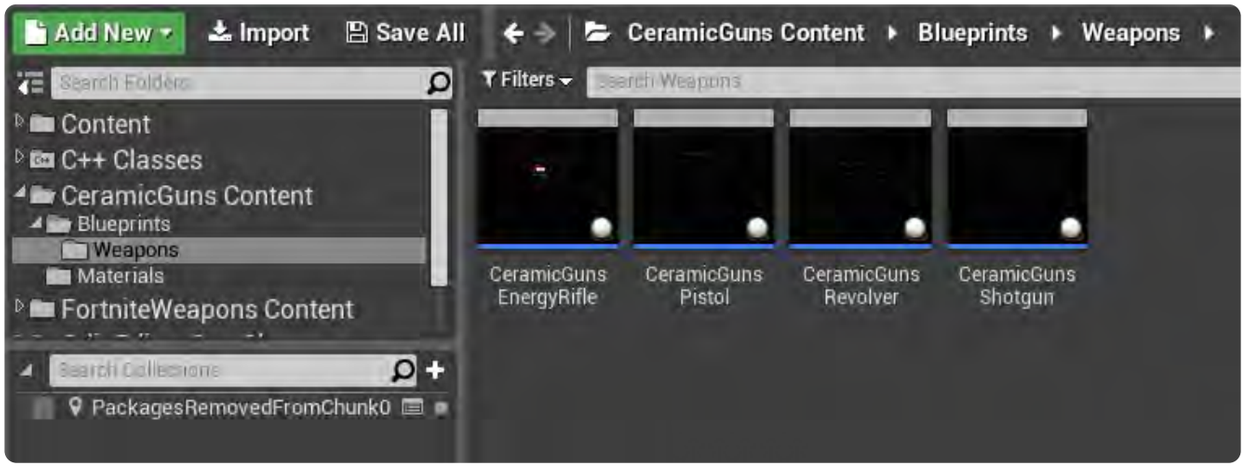
4. You can also fill out the **Author** and **Description** fields, and this data will show in the Mod menu in the game. If you'd like to change this later, you can do so by accessing your mod through the Editor's **Edit -> Plugins** menu.



5. Click the **Create Mod** button to generate your mod files.
6. Wait while the **Robo Recall Editor** creates your new mod structure, a popup tells you when it has completed.
7. Your new mod is automatically focused in the **Content Browser**.



8. Double click the **Blueprints** folder, then the **Weapons** folder to find your weapon Blueprints.



Result

With your new mod structure setup you can now start creating your mod. Now, just because you started with one mod type doesn't limit you to just modifying the couple of assets you'll find in your mod's Content Folder. You can modify any number of assets from Robo Recall and have them exist in one mod.

Just remember, that any **new** assets you import **need** to go into your mod's Content Folder. If you put them elsewhere, they will fail to package!

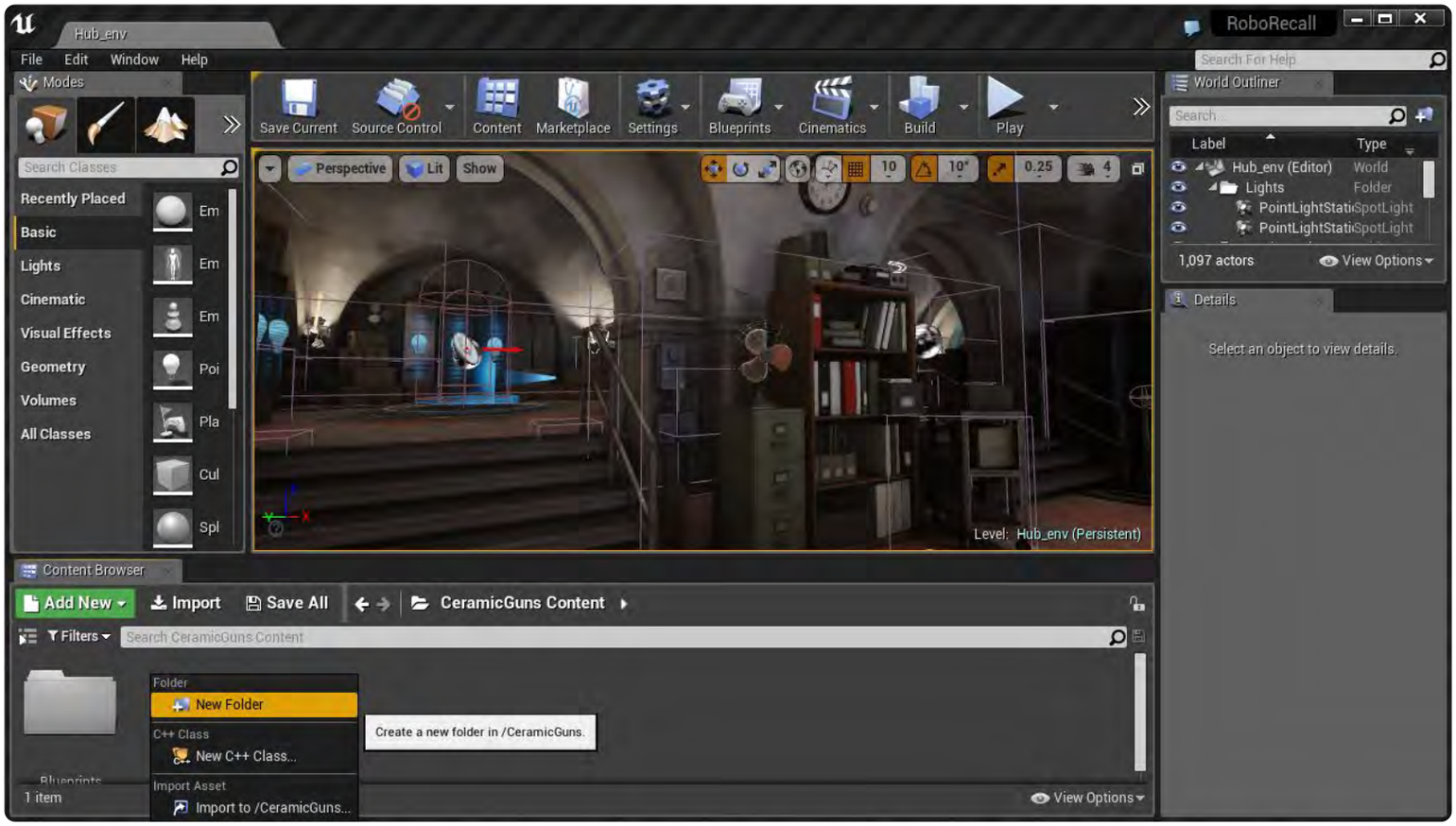
‹ Previous Step | Creating Your First Robo Recall Mod | Next Step ›

3. Add a New Weapon Material

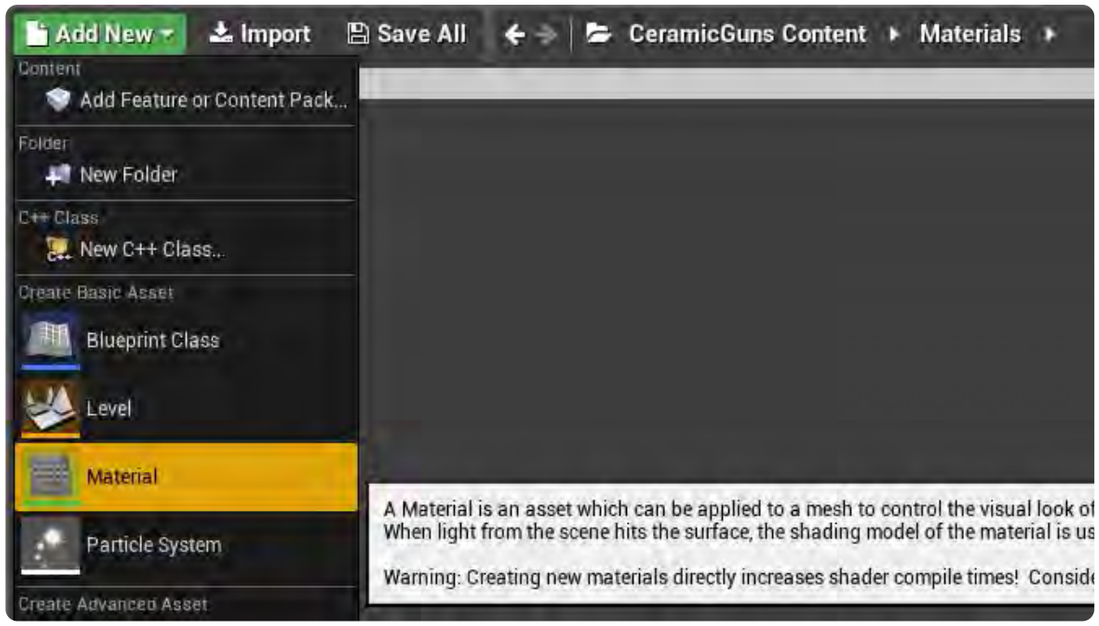
In this step, you'll be creating a new Material asset to apply to your weapons to make them look like they are fabricated from ceramic.

Steps

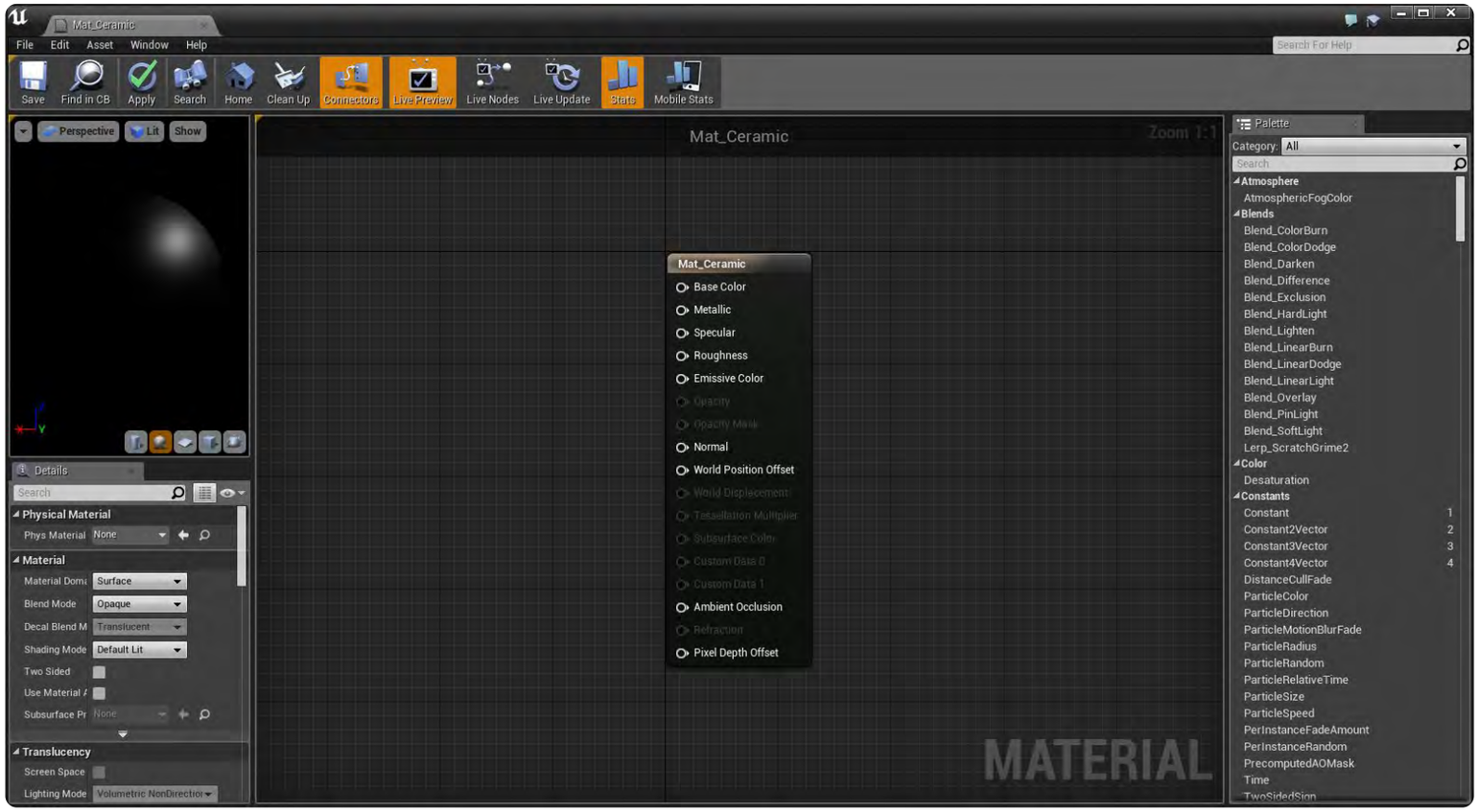
1. Right-click in the Content Browser and choose **New Folder** to create a new folder in your mod's content directory. Name the folder **Materials**.



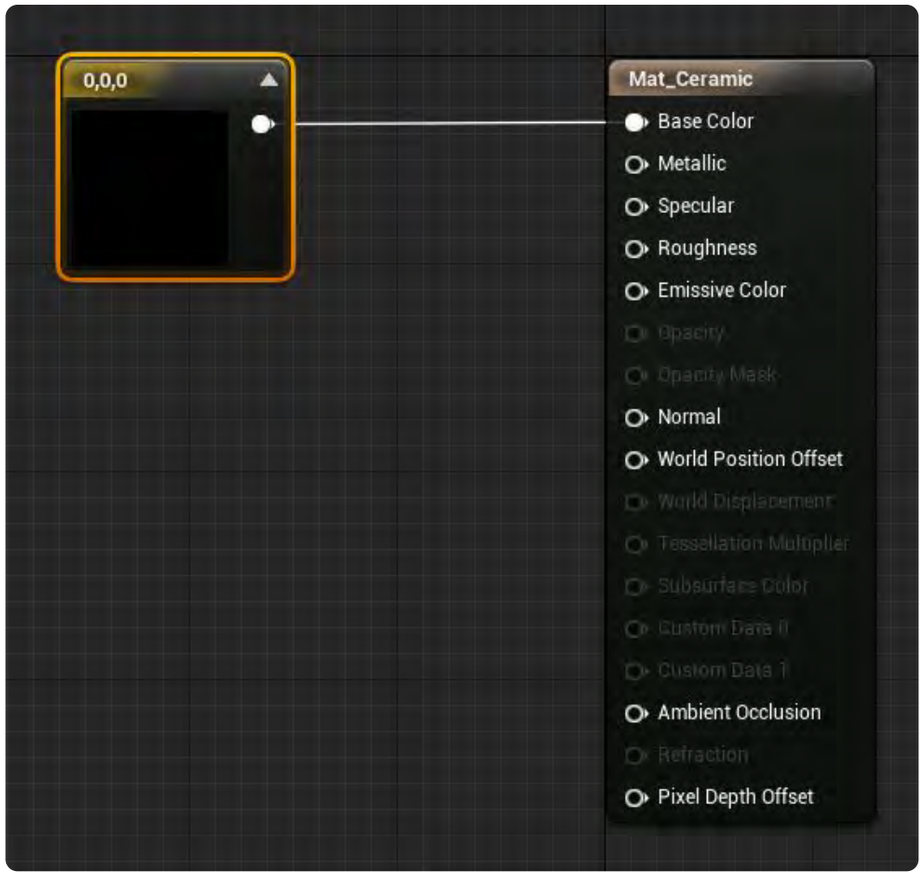
2. Double-click the Materials folder to open it and click the Add New button to add a new Material asset. Name the Material **Mat_Ceramic**.



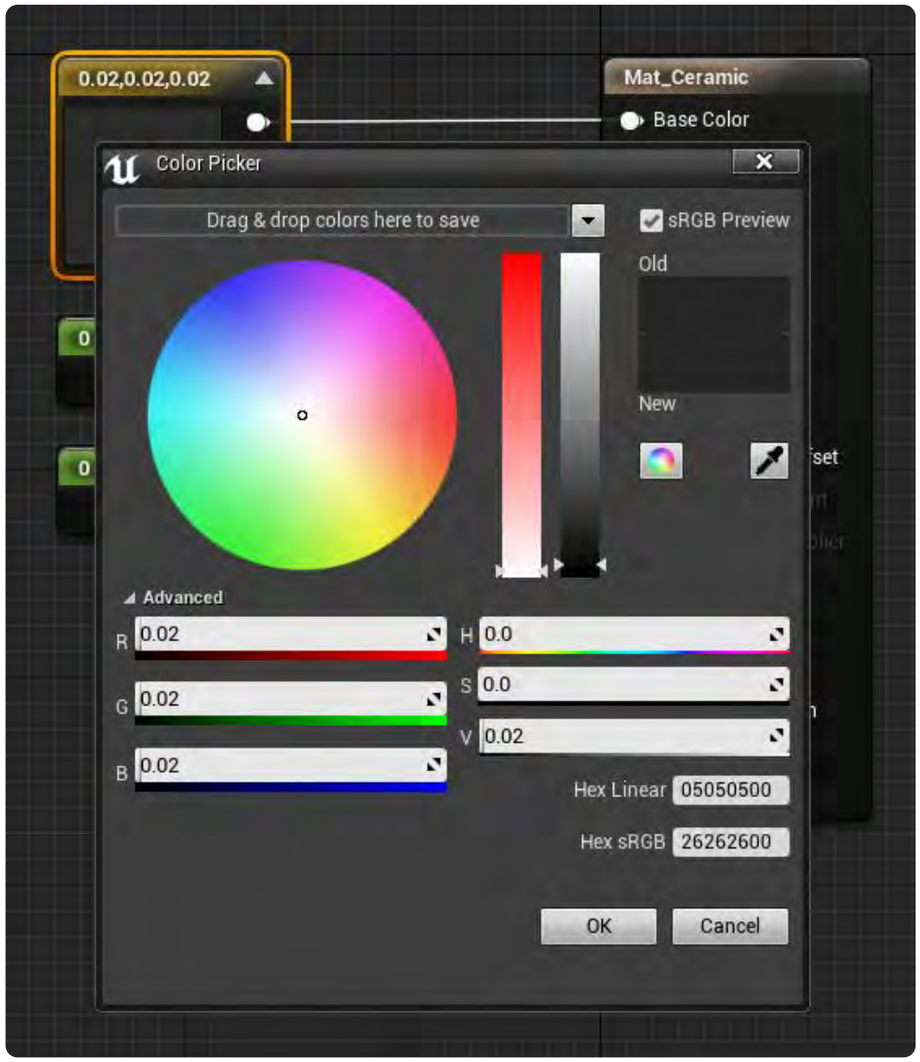
3. Double-click the *Mat_Ceramic* Material to edit it in the Material Editor .



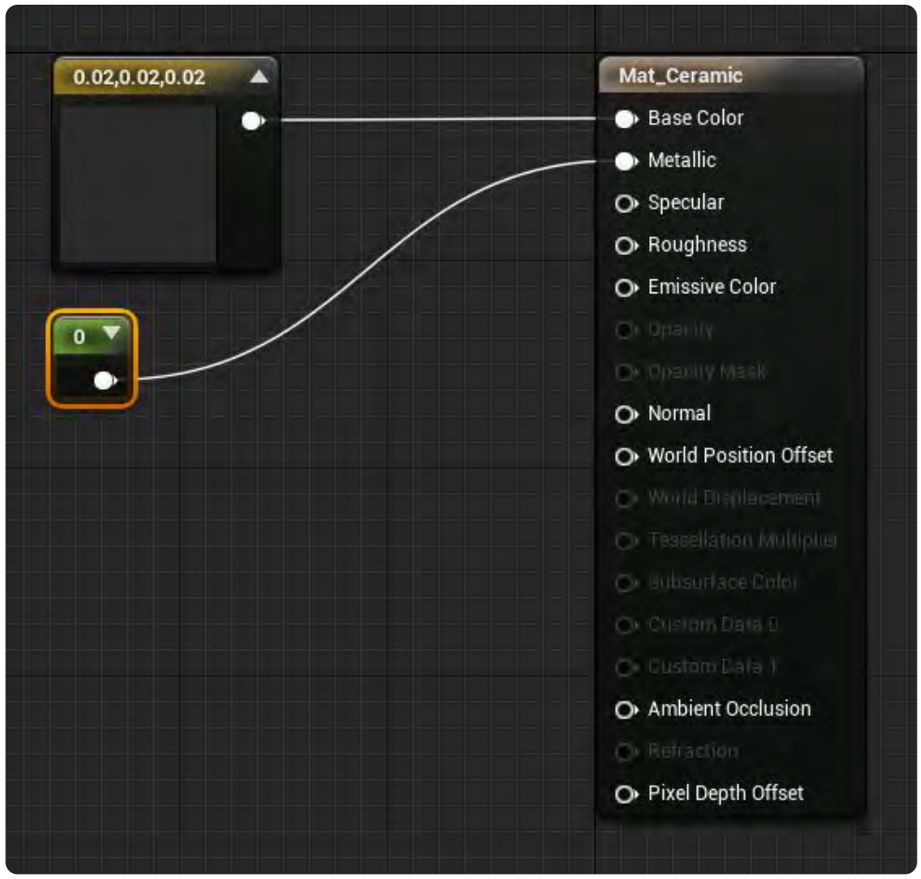
4. Drag a Constant3Vector expression into the graph from the Palette and connect it to the **Base Color** input on the Material node.



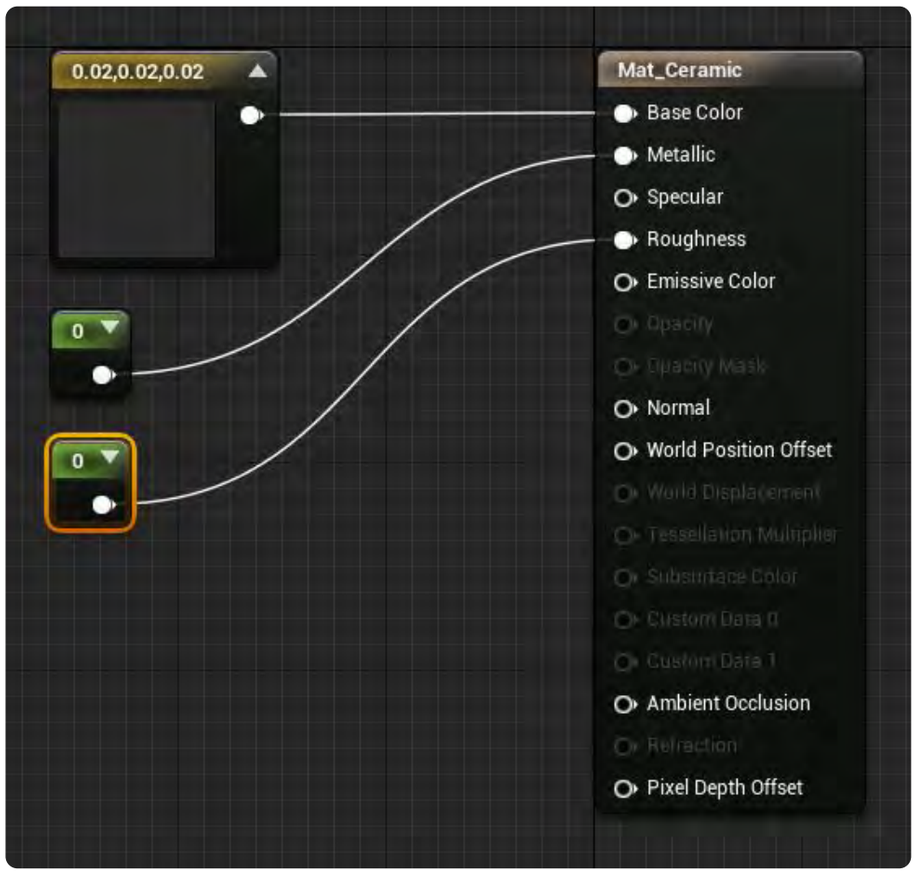
5. Double-click the black color preview on the expression to open the Color Picker . Set the R, G, and B values to 0.02 to give the surface just a tiny bit of color.



6. Drag a Constant expression into the graph from the Palette and connect it to the **Metallic** input on the Material node. Leave the value of the expression at 0. This will make the material have no metallic characteristics in its appearance.



7. Drag a Constant expression into the graph from the Palette and connect it to the Base Color input on the Material node. Leave the value of the expression at 0. This will cause the material to appear extremely shiny.



8. Click the Apply button to save the changes to the *Mat_Ceramic* Material.

Result

The Preview panel shows a dark, shiny surface that looks like ceramic. In the next step, you will apply this Material to your modded weapons to give them a ceramic appearance.

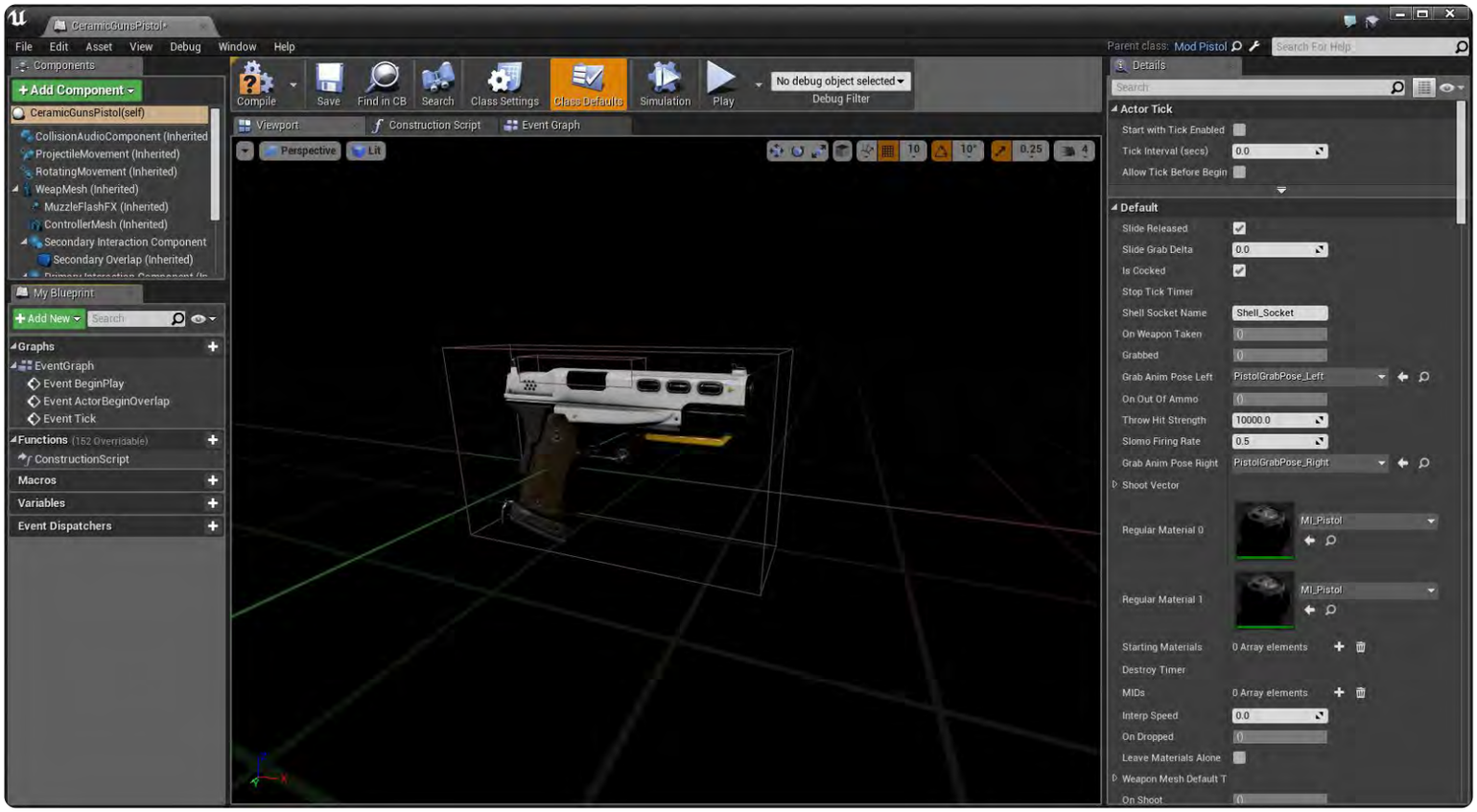
Previous Step
Creating Your First Robo Recall Mod
Next Step

4. Apply the Weapon Material

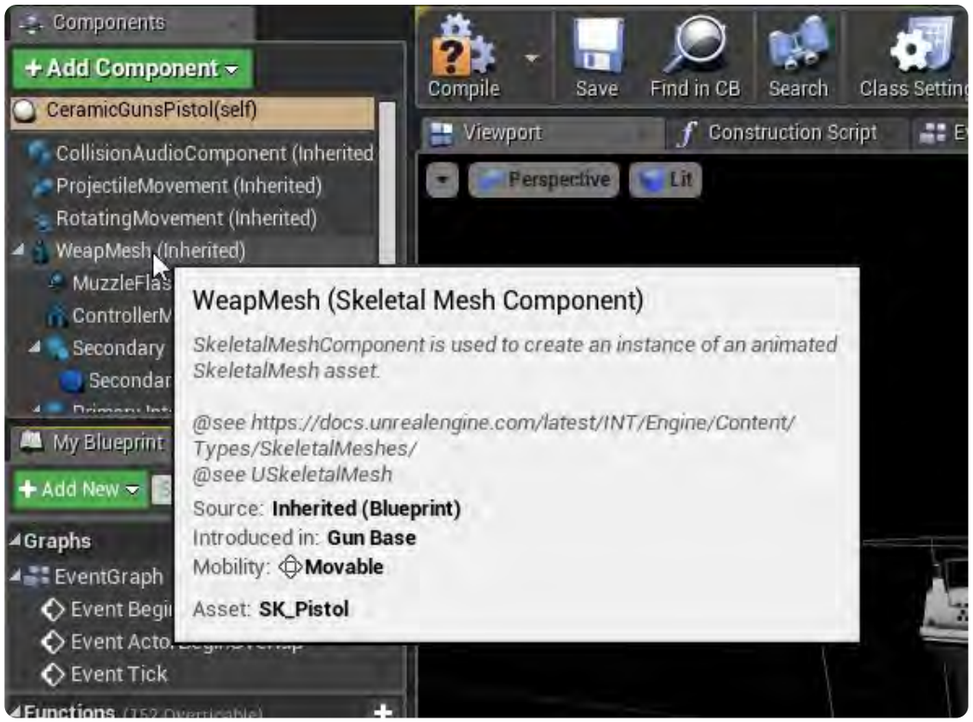
It's time to apply your ceramic Material to your weapons so they look ceramic instead of metallic.

Steps

1. Double-click one of your weapon Blueprints in the Content Browser to edit it in the Blueprint Editor .

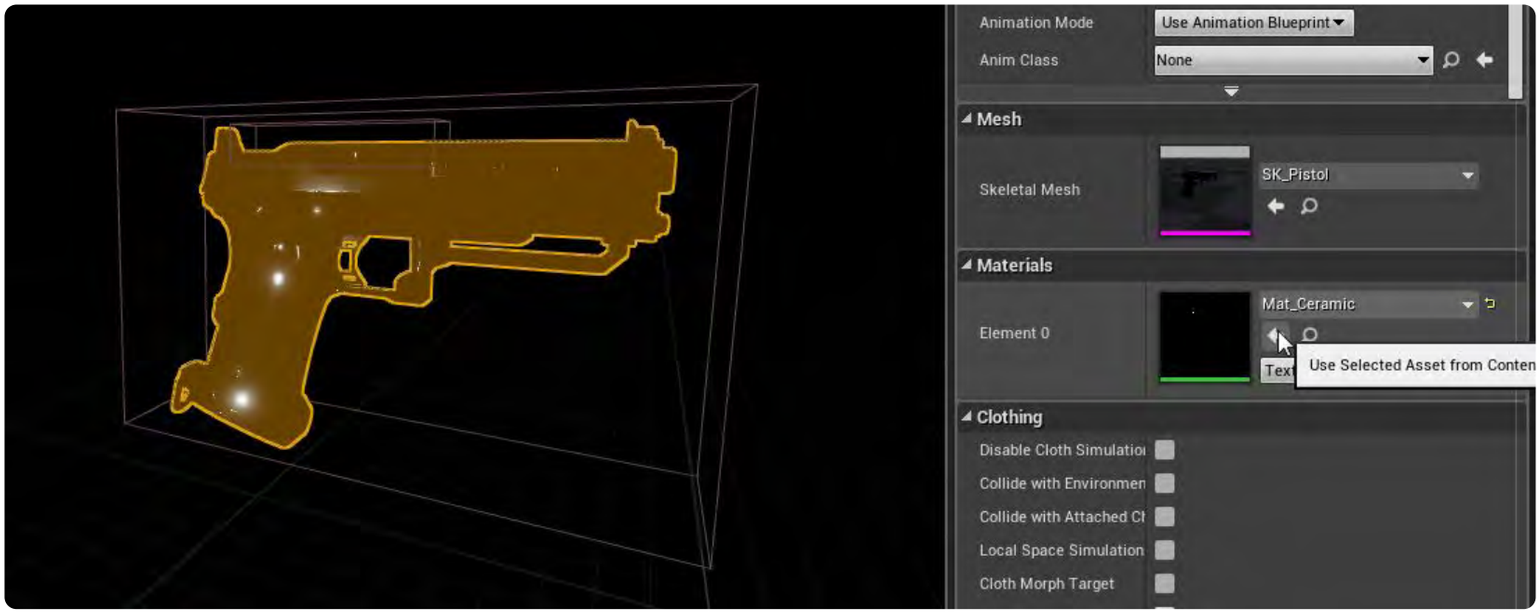


2. Select the **WeapMesh** Component in the Components panel.

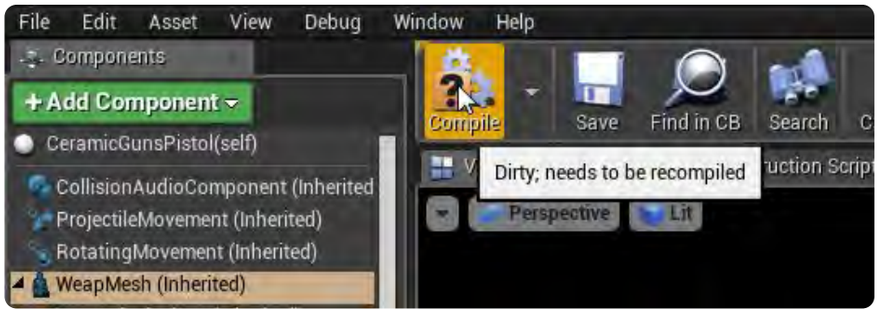


3. Back in the Content Browser, select the *Mat_Ceramic* Material.

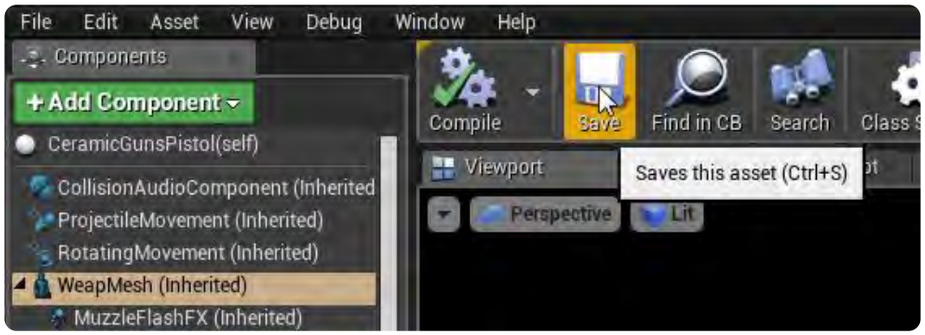
4. In the Details panel, find the **Materials** category and click the **Use Selected Asset from Content BRowser** button to apply the *Mat_Ceramic* Material to the weapon mesh.



5. Click the **Compile** button in the Blueprint Editor toolbar to update the Blueprint with the changes.



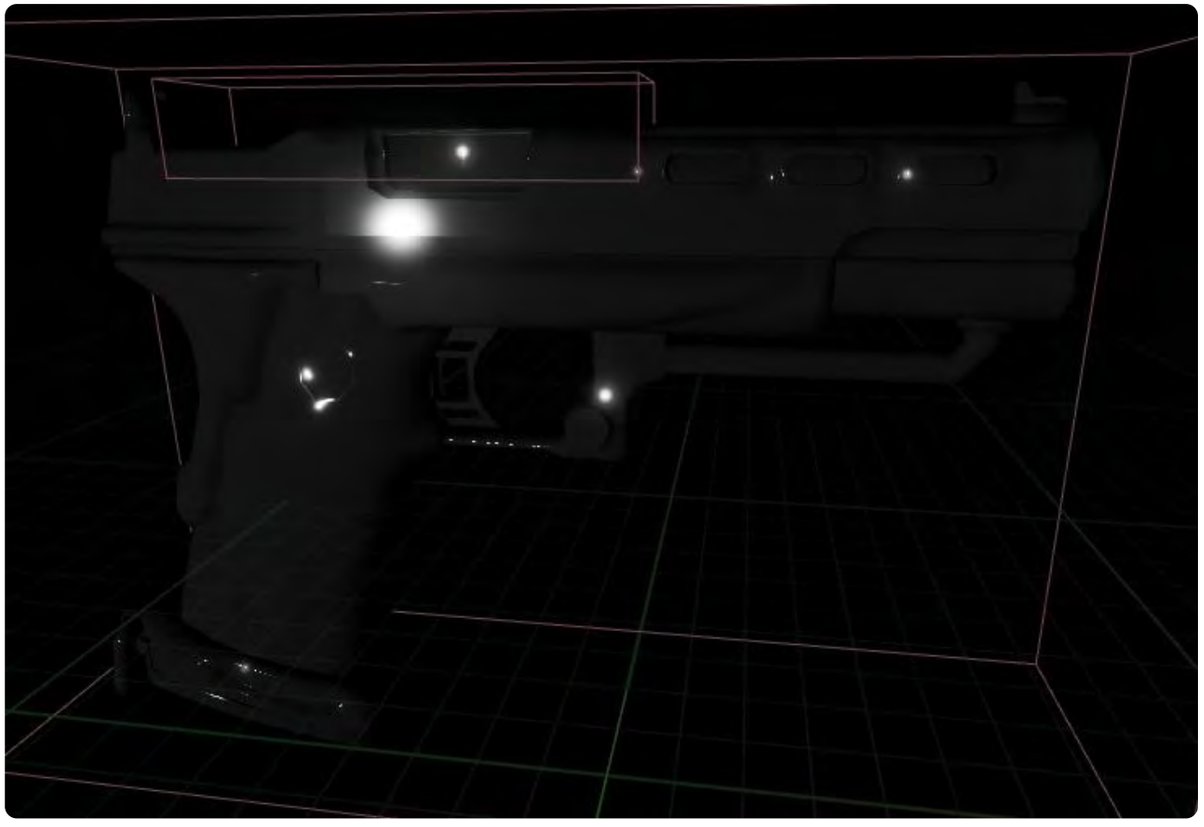
6. Click the **Save** button in the Blueprint Editor toolbar to save the weapon asset.



7. Repeat these steps for the other weapon Blueprints in your mod to make them all ceramic.

Result

The Preview panel shows the ceramic weapon. In the next step, you will test out the mod to see your ceramic weapons in action!



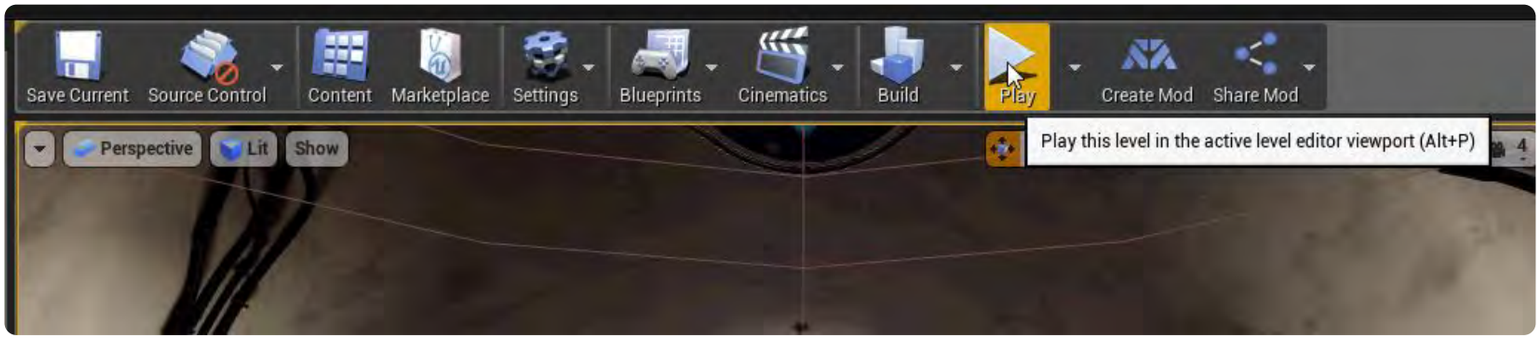
‹ Previous Step | Creating Your First Robo Recall Mod | Next Step ›

5. Test Your Mod

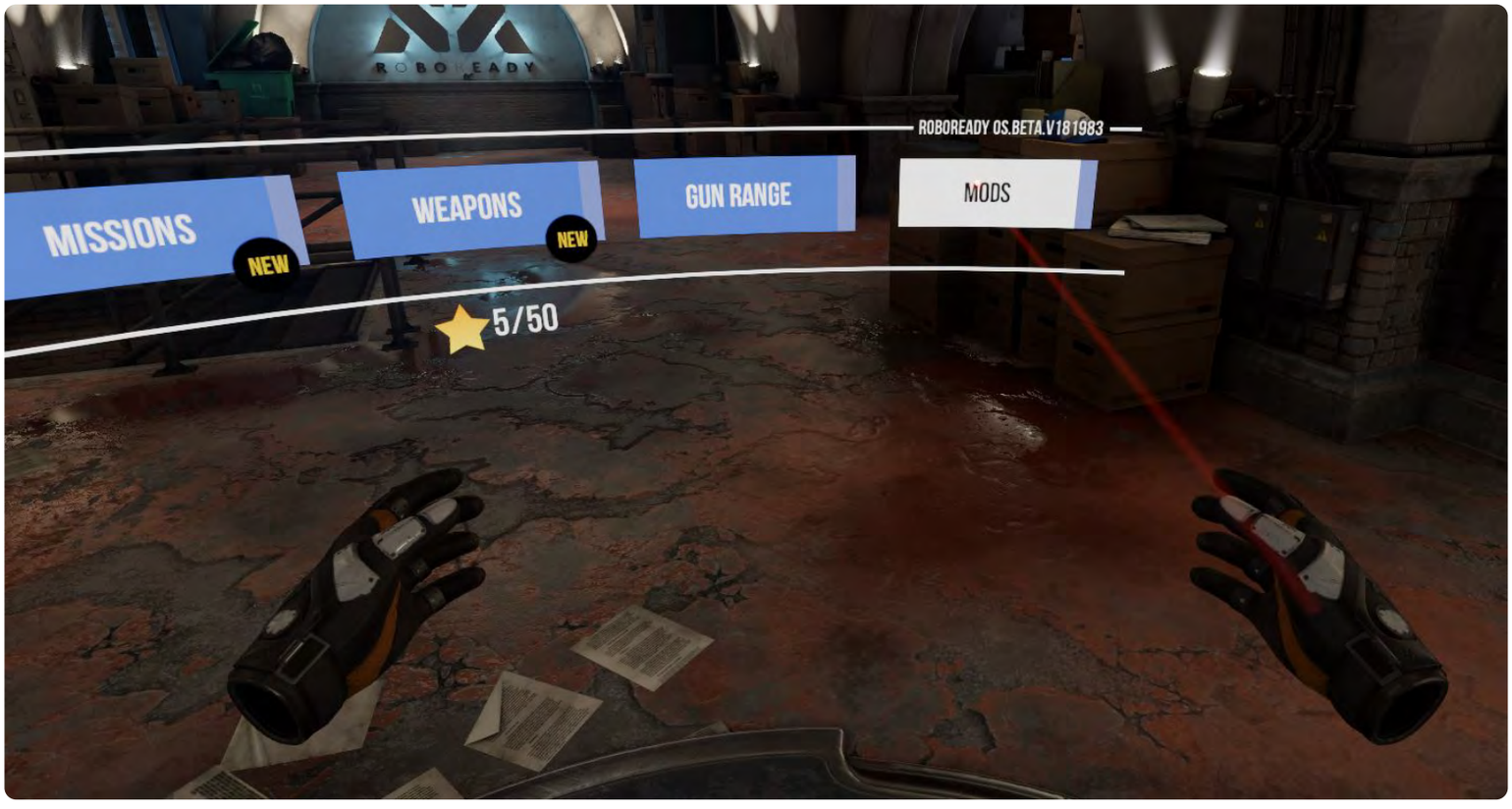
Let's make sure our mod is working as intended by enabling it int he game and shooting some targets on the gun range.

Steps

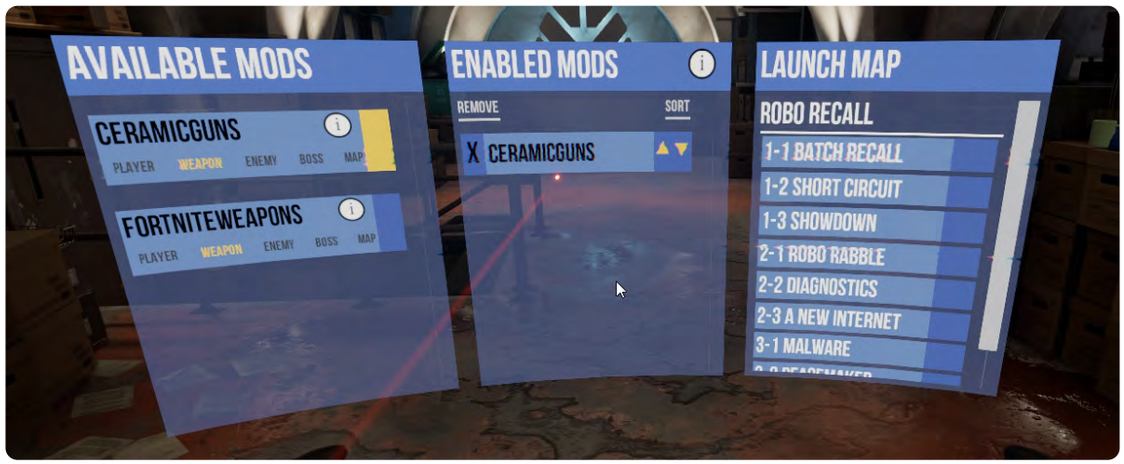
1. In the Level Editor toolbar, click the **Play** button to launch into the Hub.



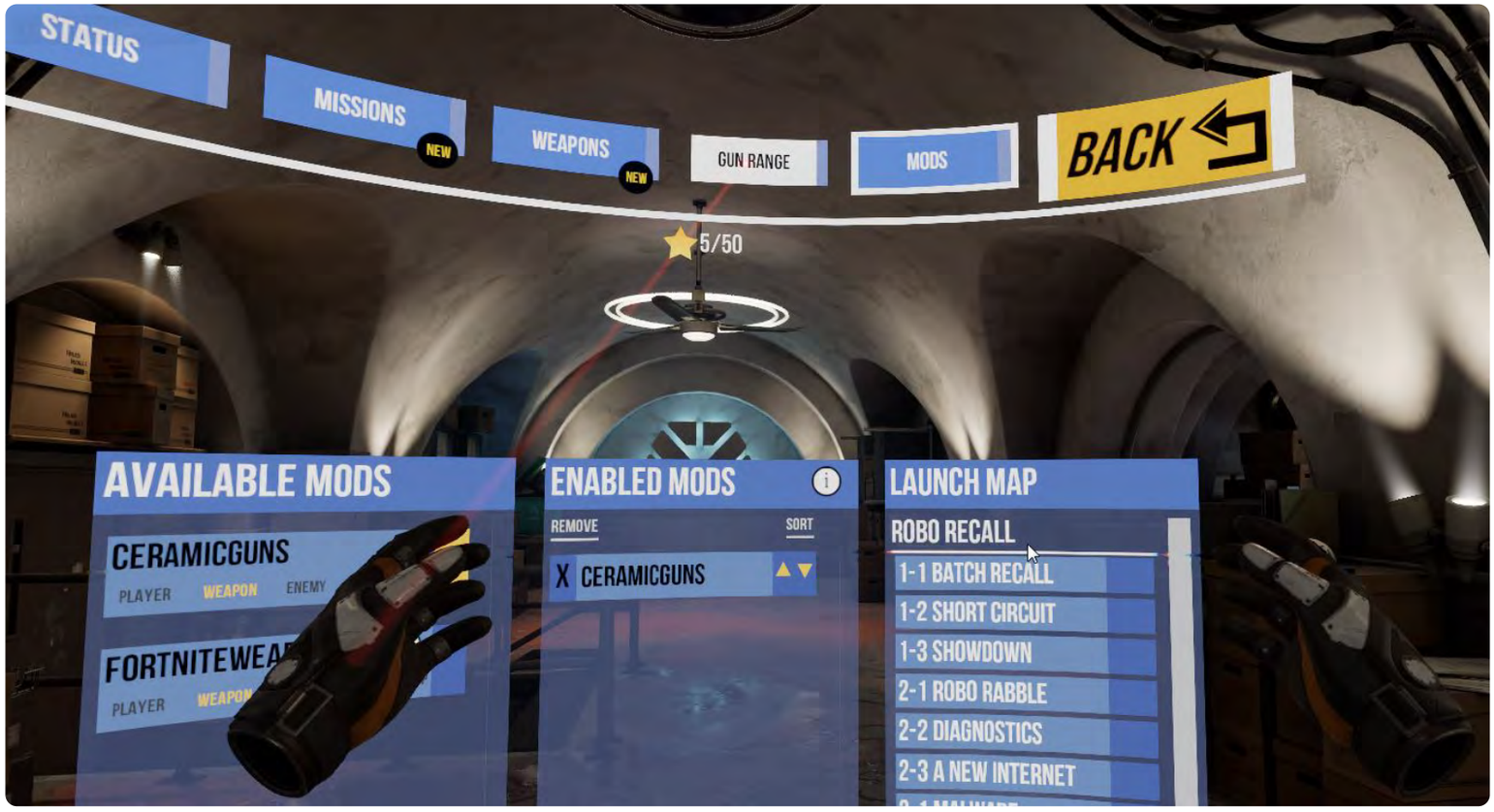
2. Teleport to the holostation to open up the menu and select **MODS**.



3. In the Mods menu, find the **Ceramic Guns** mod and select it to enable it.



4. Select **GUN RANGE** from the main menu to do some target practice.



5. Your ceramic guns are available to be equipped.

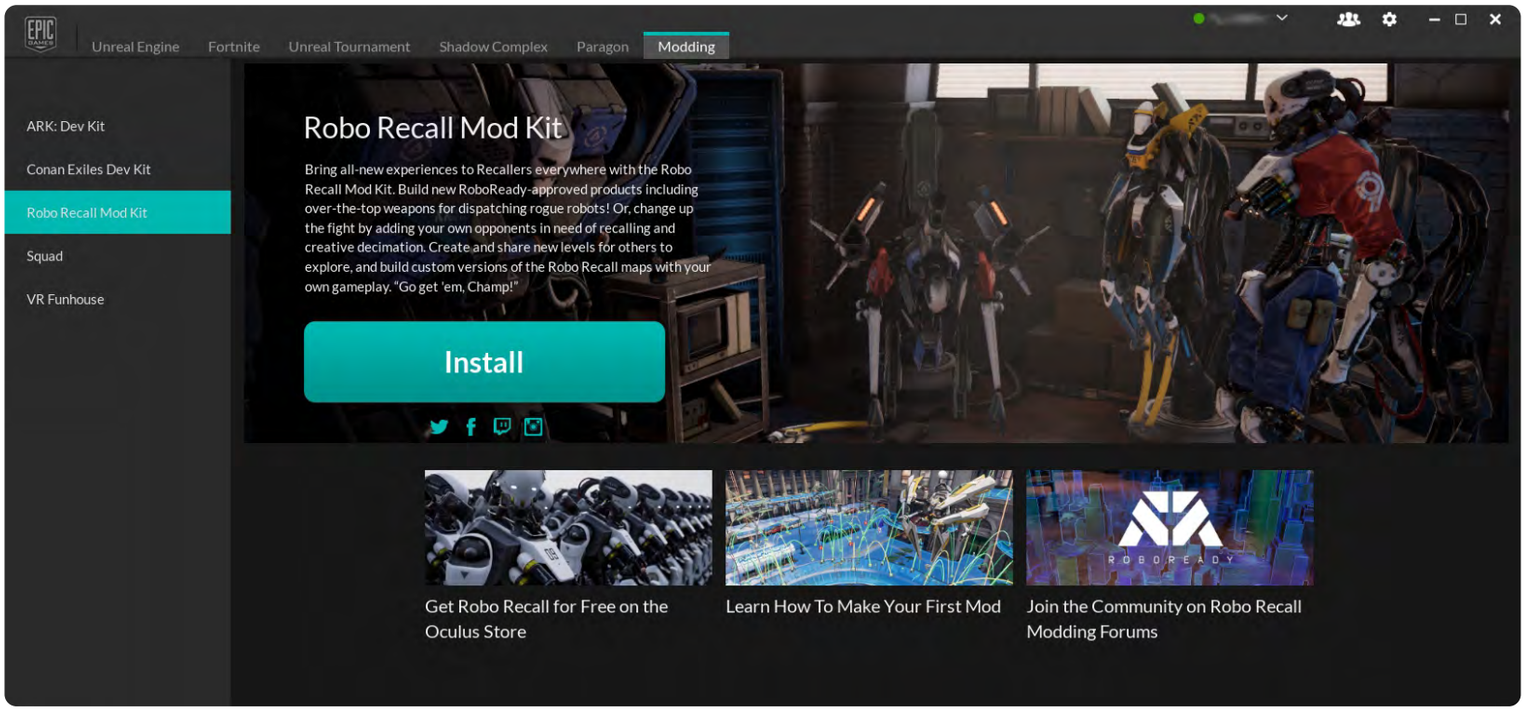
Result

Shoot some targets with your new ceramic weapons!

Previous Step
Creating Your First Robo Recall Mod
Next Step

Installing the Robo Recall Mod Kit

1. Sign up for an Epic ID
2. Download and install the Epic Games Launcher
3. Click **Install** to download the Robo Recall Mod Kit from the Epic Games Launcher.

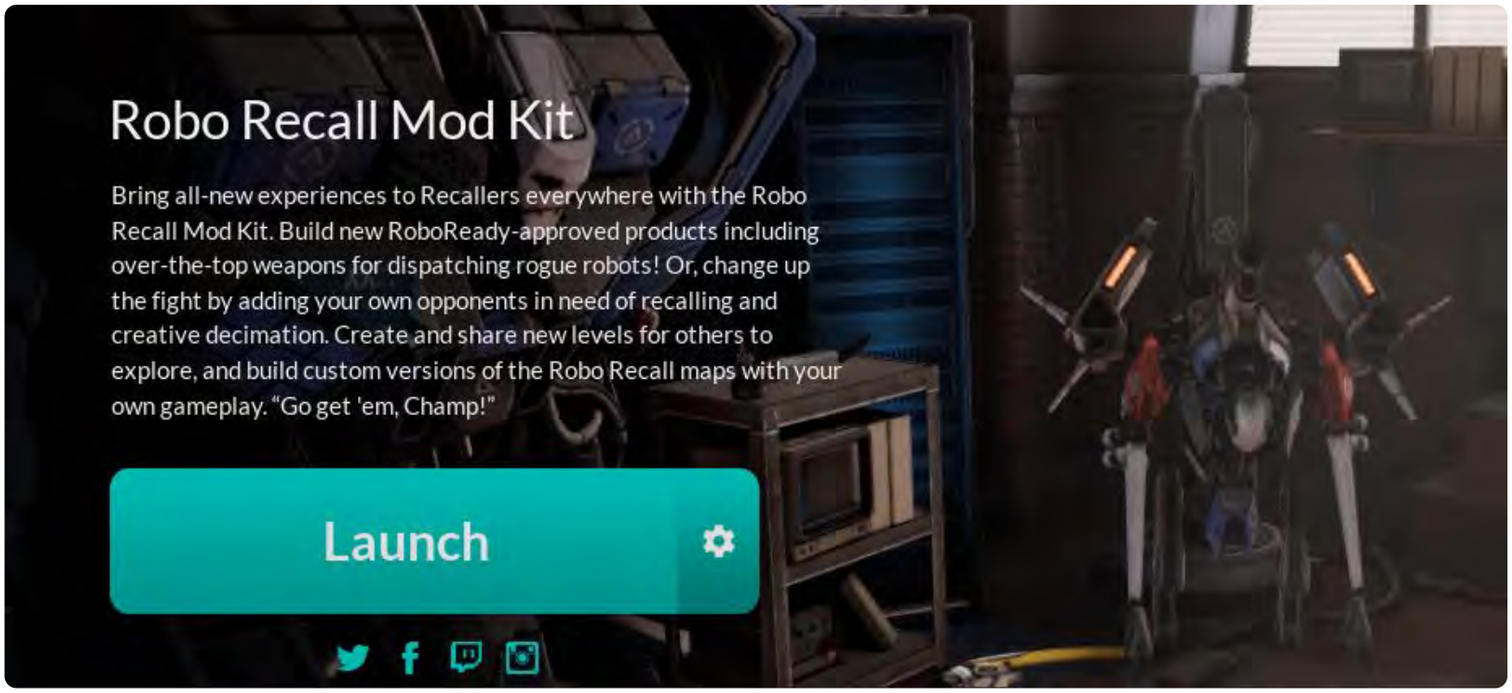


This can take some time as the Robo Recall Editor is quite large (20+ GB). Take this time to review some of the other Robo Recall Documentation or perhaps some of the Unreal Engine 4 getting started guides:

- **Level Designer Quick Start** ›
- **Blueprints Visual Scripting** ›
- **Programming Quick Start** ›
- **Managing Content** ›

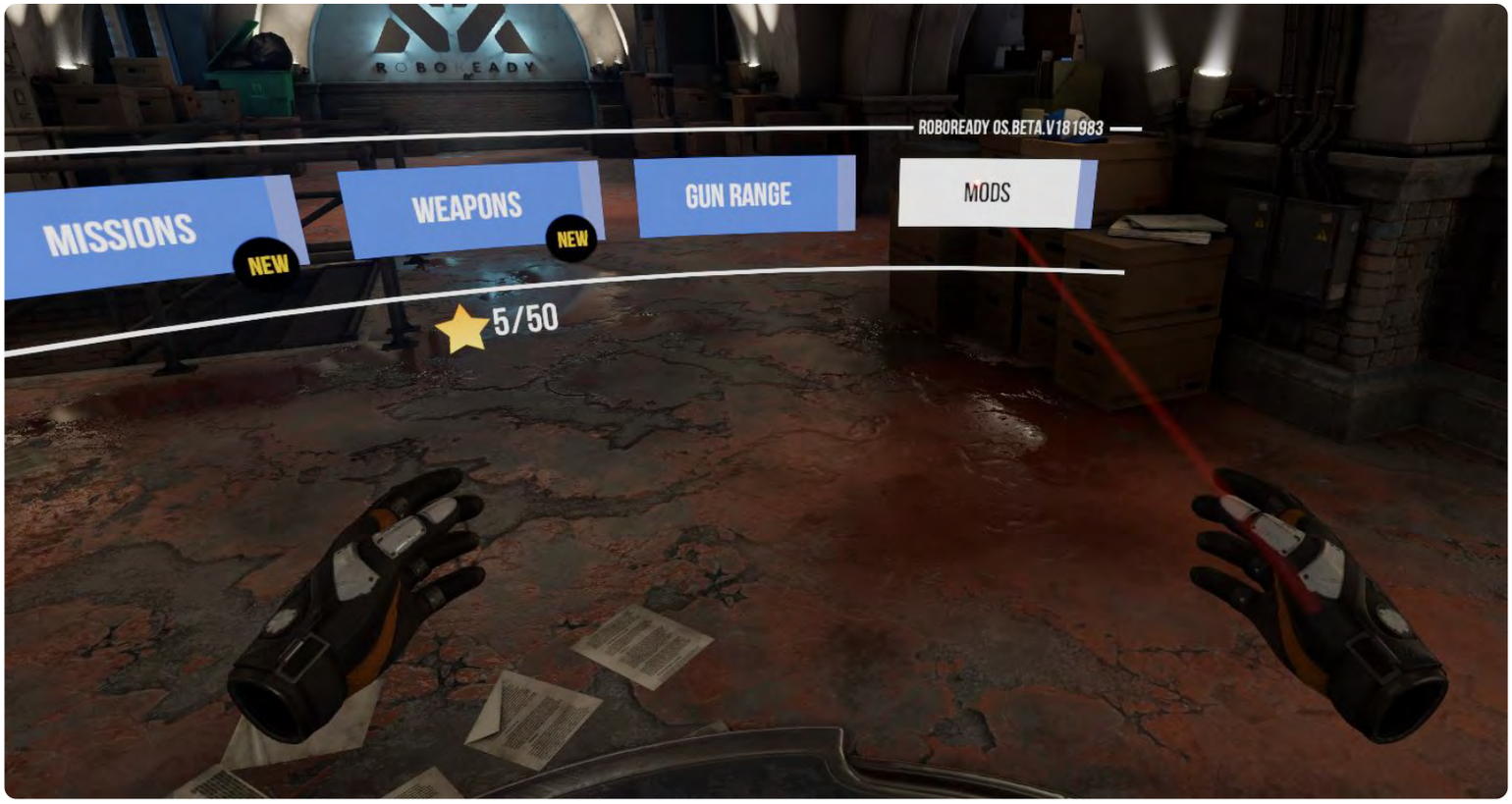
Once the download is complete, head to the next step.

4. Launch the Robo Recall Editor.

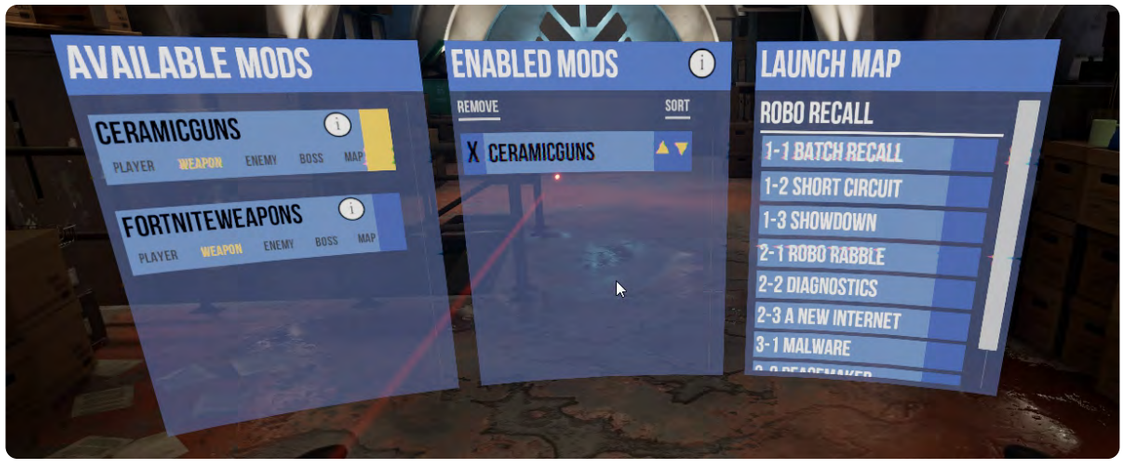

Robo Recall Mod Kit
Bring all-new experiences to Recallers everywhere with the Robo Recall Mod Kit. Build new RoboReady-approved products including over-the-top weapons for dispatching rogue robots! Or, change up the fight by adding your own opponents in need of recalling and creative decimation. Create and share new levels for others to explore, and build custom versions of the Robo Recall maps with your own gameplay. “Go get 'em, Champ!”
Launch

Enabling a Mod

1. Teleport to the holostation to open up the menu and select **MODS**.



2. In the Mods menu, find the mod you want to enable and select it to enable it.


AVAILABLE MODS
CERAMICGUNS
PLAYER
WEAPON
ENEMY
BOSS
FORTNITEWEAPONS
PLAYER
WEAPON
ENEMY
BOSS
MAP
ENABLED MODS
REMOVE
SORT
X CERAMICGUNS
LAUNCH MAP
ROBO RECALL
1-1 BATCH RECALL
1-2 SHORT CIRCUIT
1-3 SHOWDOWN
2-1 ROBO RABBLE
2-2 DIAGNOSTICS
2-3 A NEW INTERNET
3-1 MALWARE

Overriding and Extending Functionality

[COMMENT:none] Location: Modding/RoboRecall/Quickstarts/WeaponMod [/COMMENT]

In this guide, you will make a new weapon mod using the Pistol as a base which you will modify to explode like a grenade when thrown into the world by overriding some of the existing functionality.

Steps

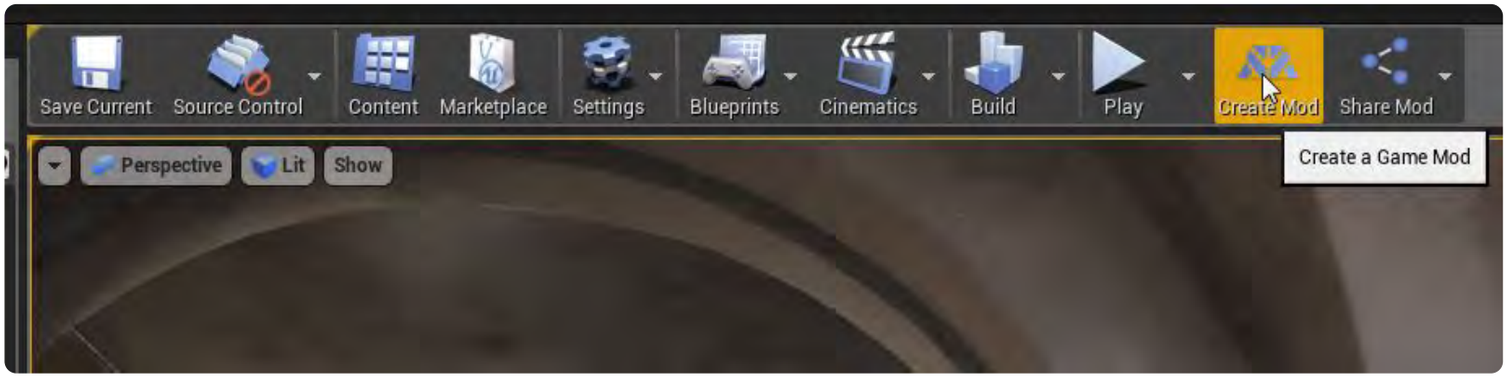
Click to Start

1. Creating a New Pistol Mod

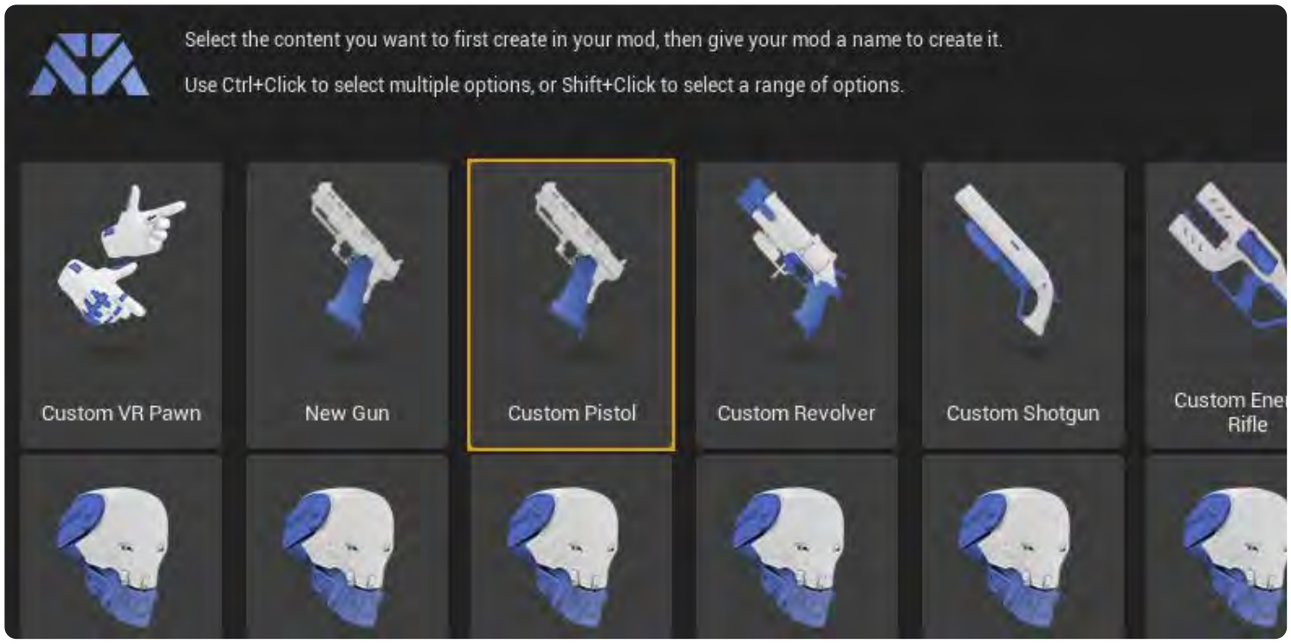
Previous Step | Overriding and Extending Functionality | Next Step

Steps

1. Click the **Create Mod** button on the Toolbar.

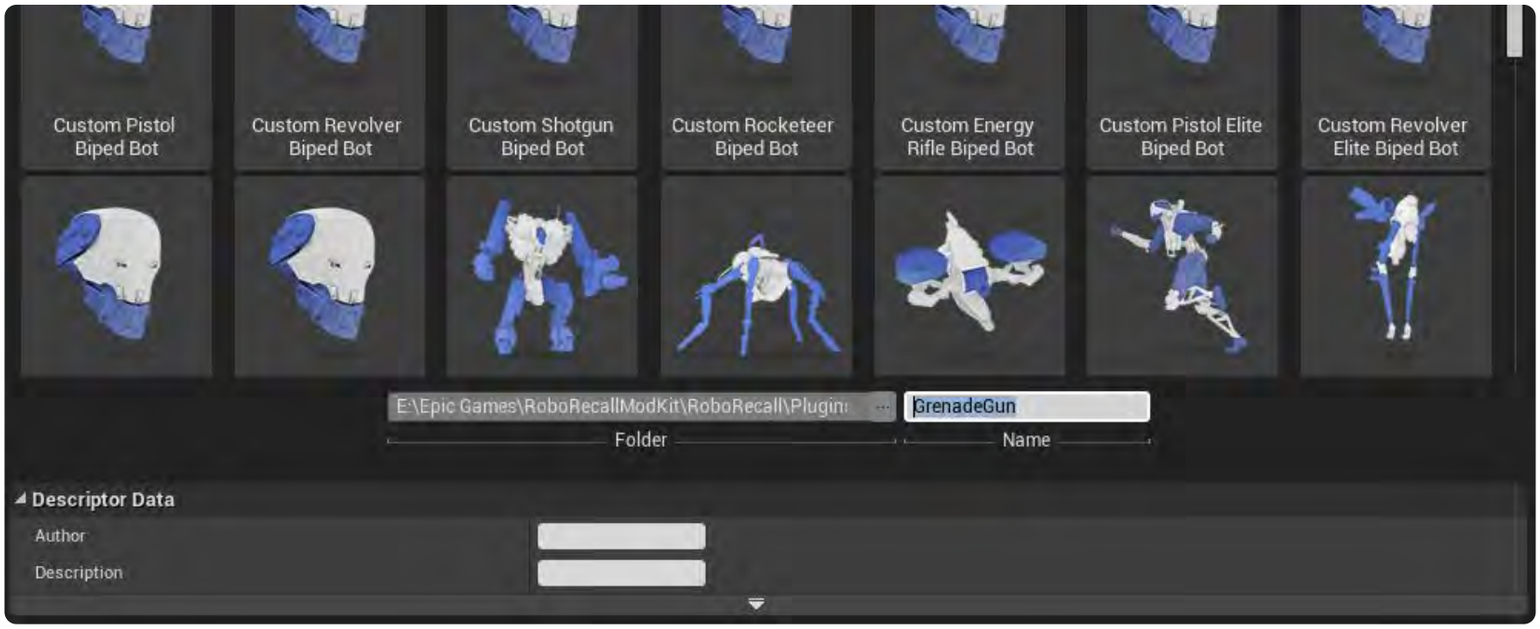


2. Select the **Custom Pistol** from hte available mod types.

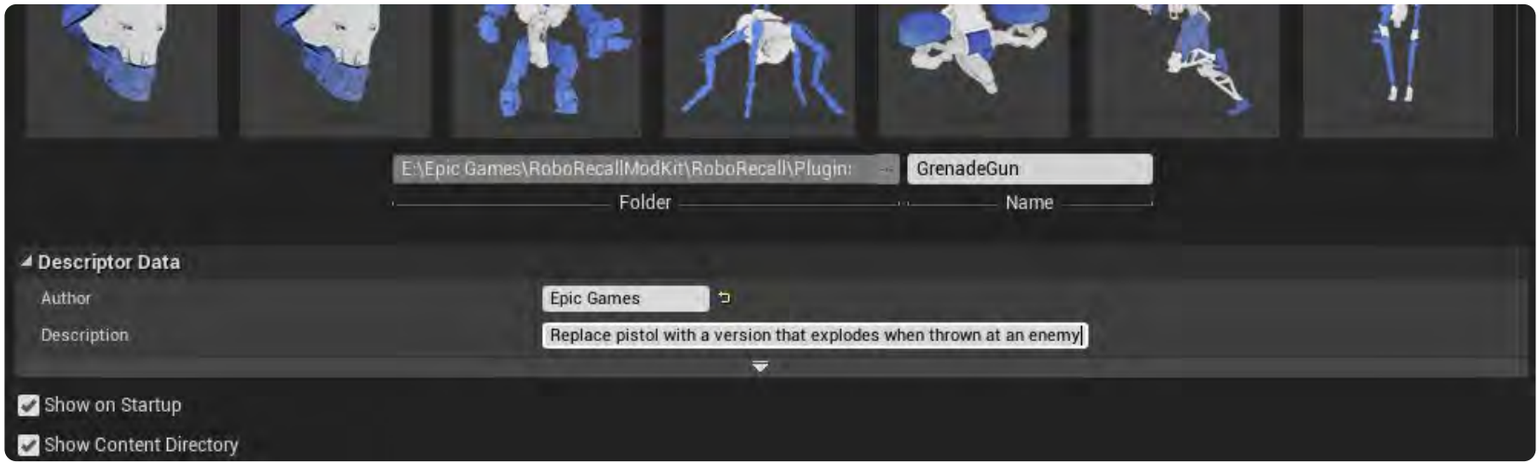


> You can multiselect in this dialog to create multiple assets at the same time. Feel free to add any other weapons, bots, etc. by **Ctrl + Clicking** the assets you want to create in your mod.

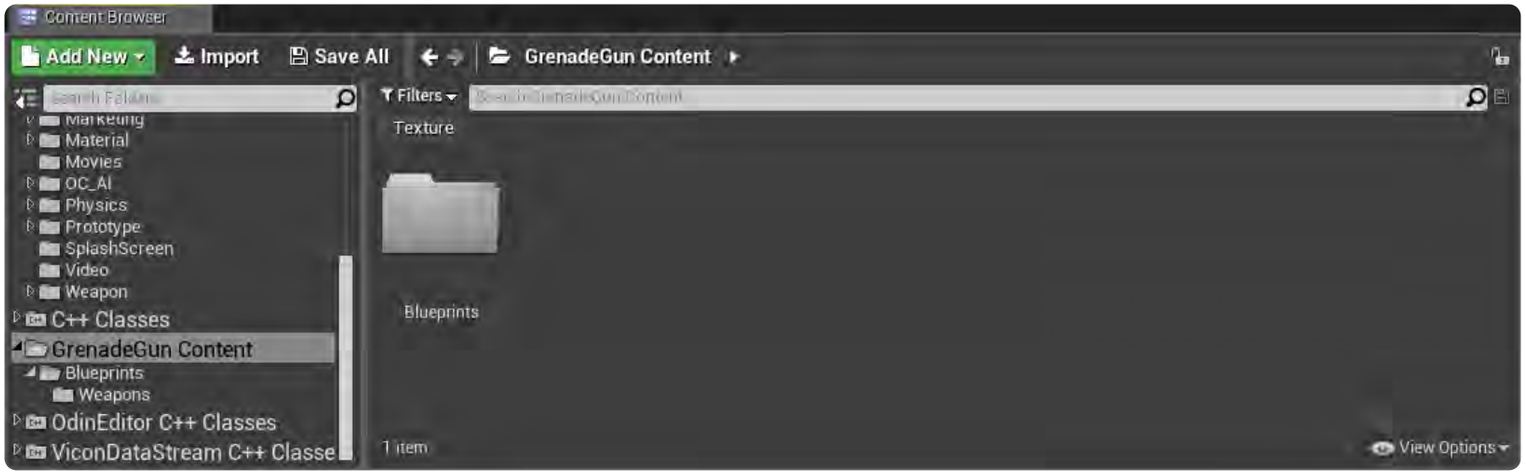
3. Enter a name for the mod in the **Name** field. In this case, we've chosen **GrenadeGun**.



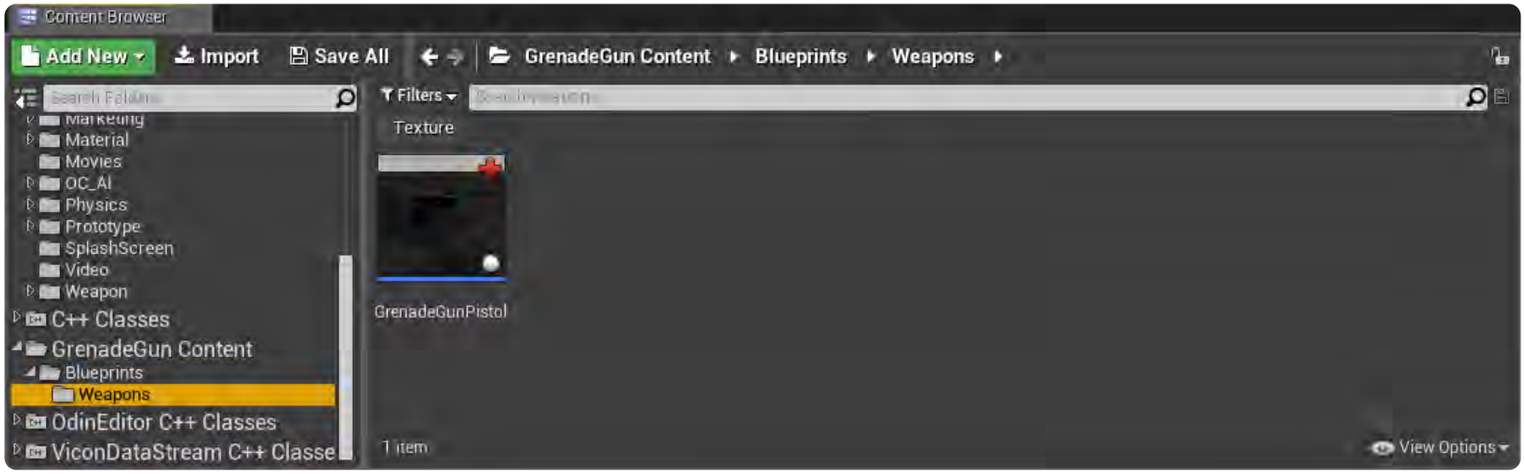
4. You can enter an **Author** and **Description**, and this data will show in the Mod menu in the game. If you'd like to change this later, you can do so by accessing your mod through the Editor's **Edit -> Plugins** menu.



5. Click the **Create Mod** button.
6. Your new mod package will be automatically focused in the **Content Browser**, but it will be at the root level of the mod.

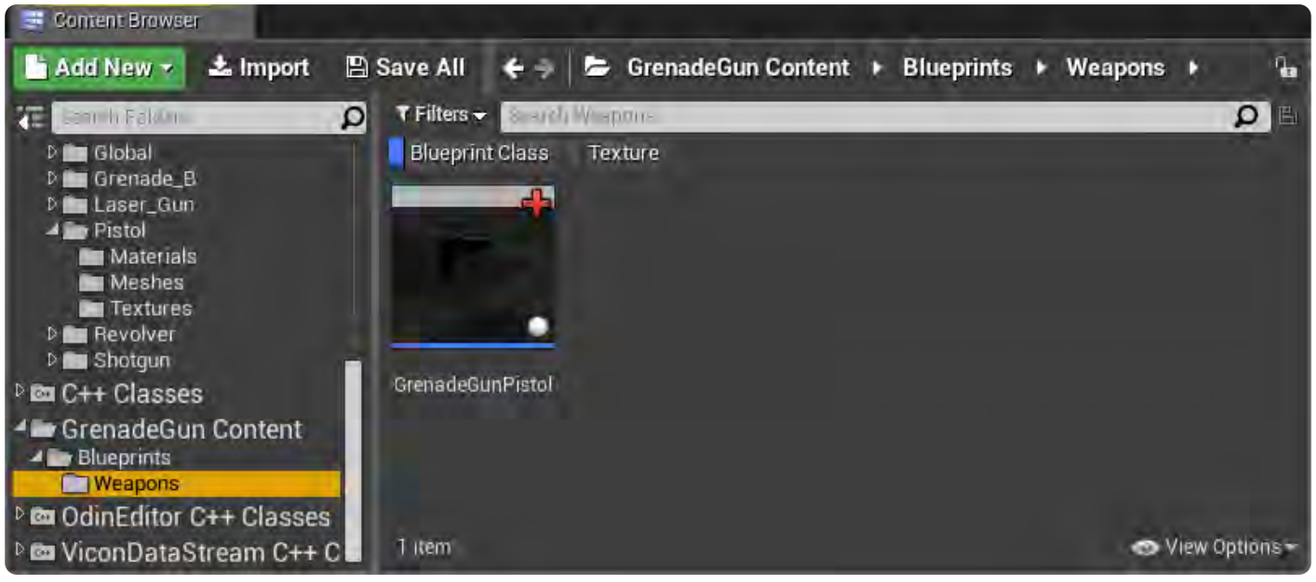


7. Double click the **Blueprints** folder, then the **Weapons** folder to find the Blueprint asset for your new custom pistol.



Results

You now have a mod with a custom pistol that will replace the default pistol in the game. Now it needs the functionality to cause it to explode when thrown.



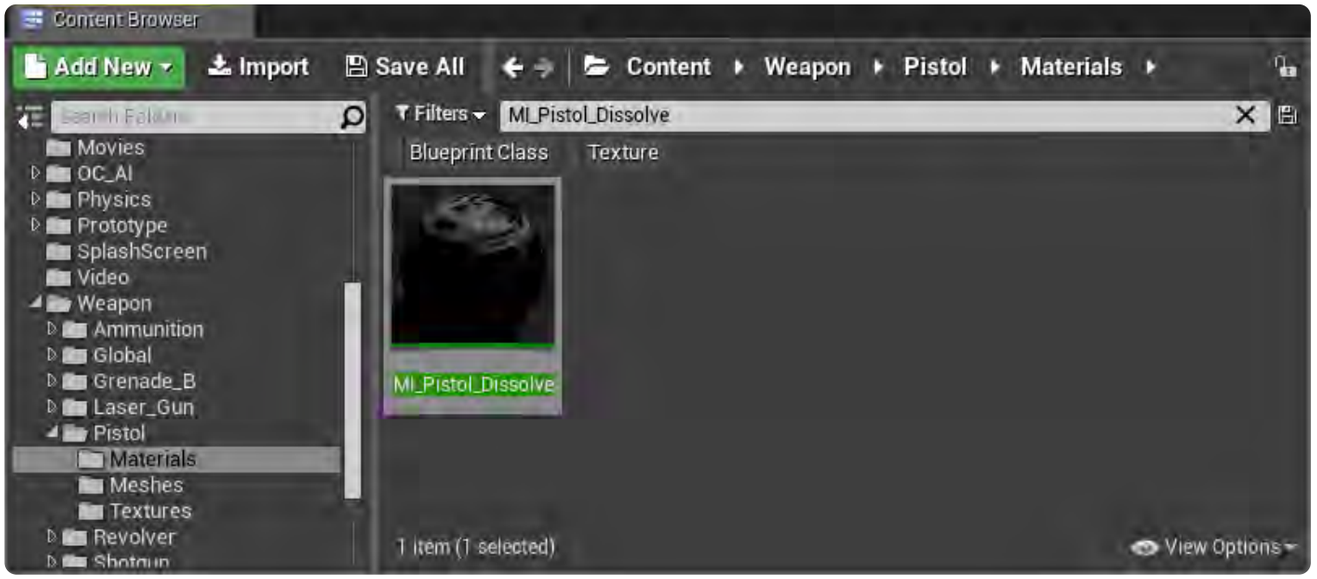
Previous Step | Overriding and Extending Functionality | Next Step

2. Creating a New Material Instance

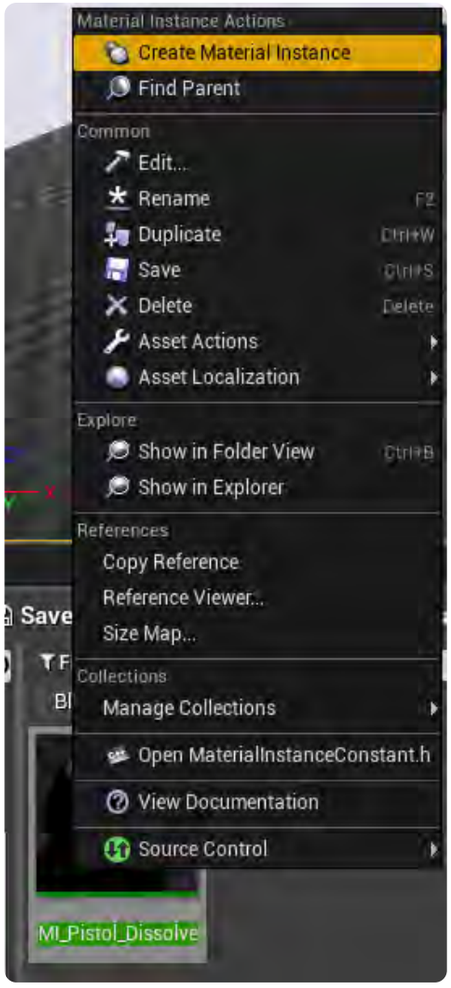
You have a custom pistol, but it still behaves like the default pistol. When thrown, it has a "dissolve" effect that is blue. Since our pistol is going to explode, a red effect would make more sense. To set that up, you need a new Material Instance of the "dissolving" material used by the **Pistol**.

Steps

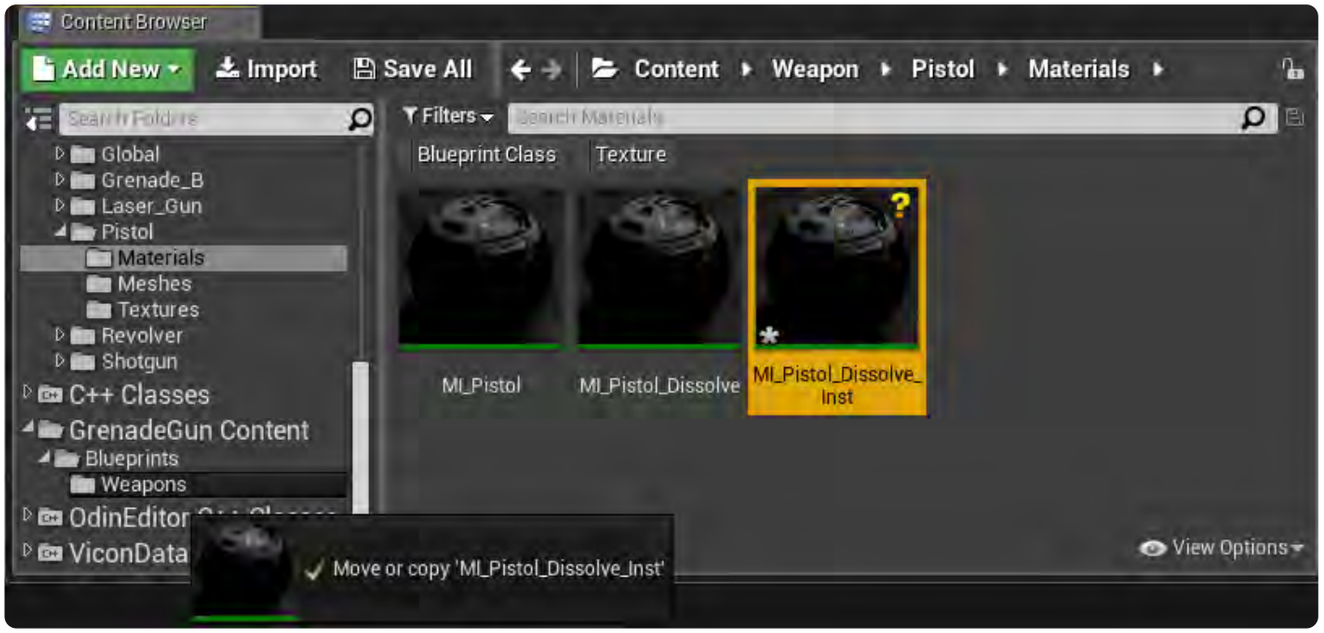
1. Search for **MI_Pistol_Dissolve** in the **Content Browser** using the search box.



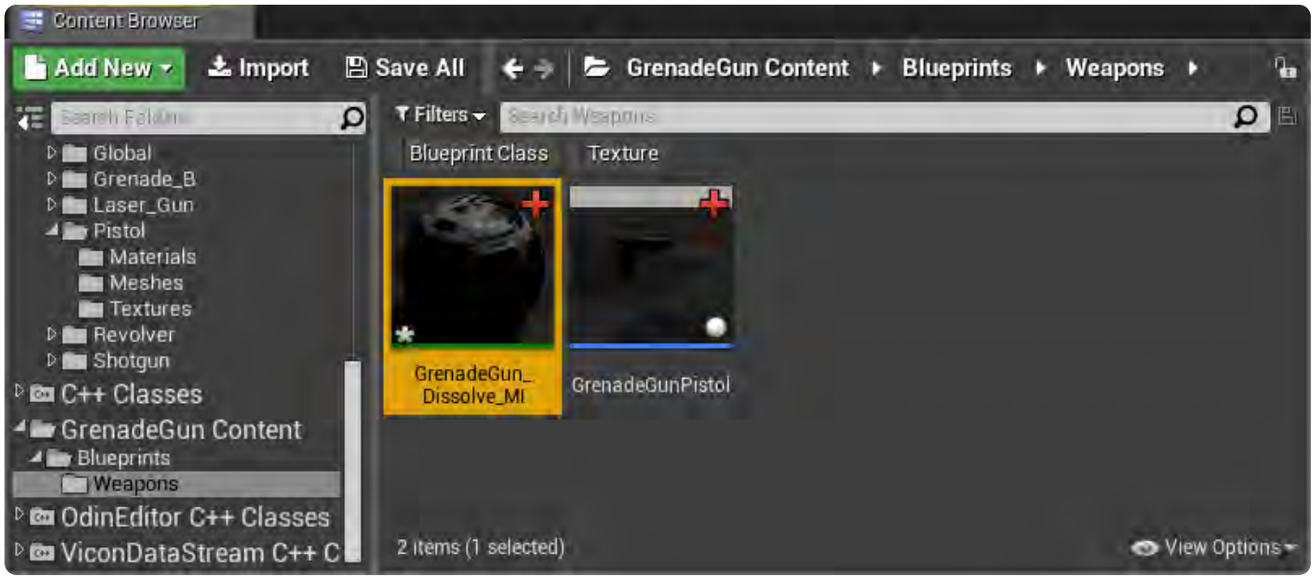
2. **Right-Click** the asset and choose **Create Material Instance** to create a new Material Instance based off *MI_Pistol_Dissolve*.



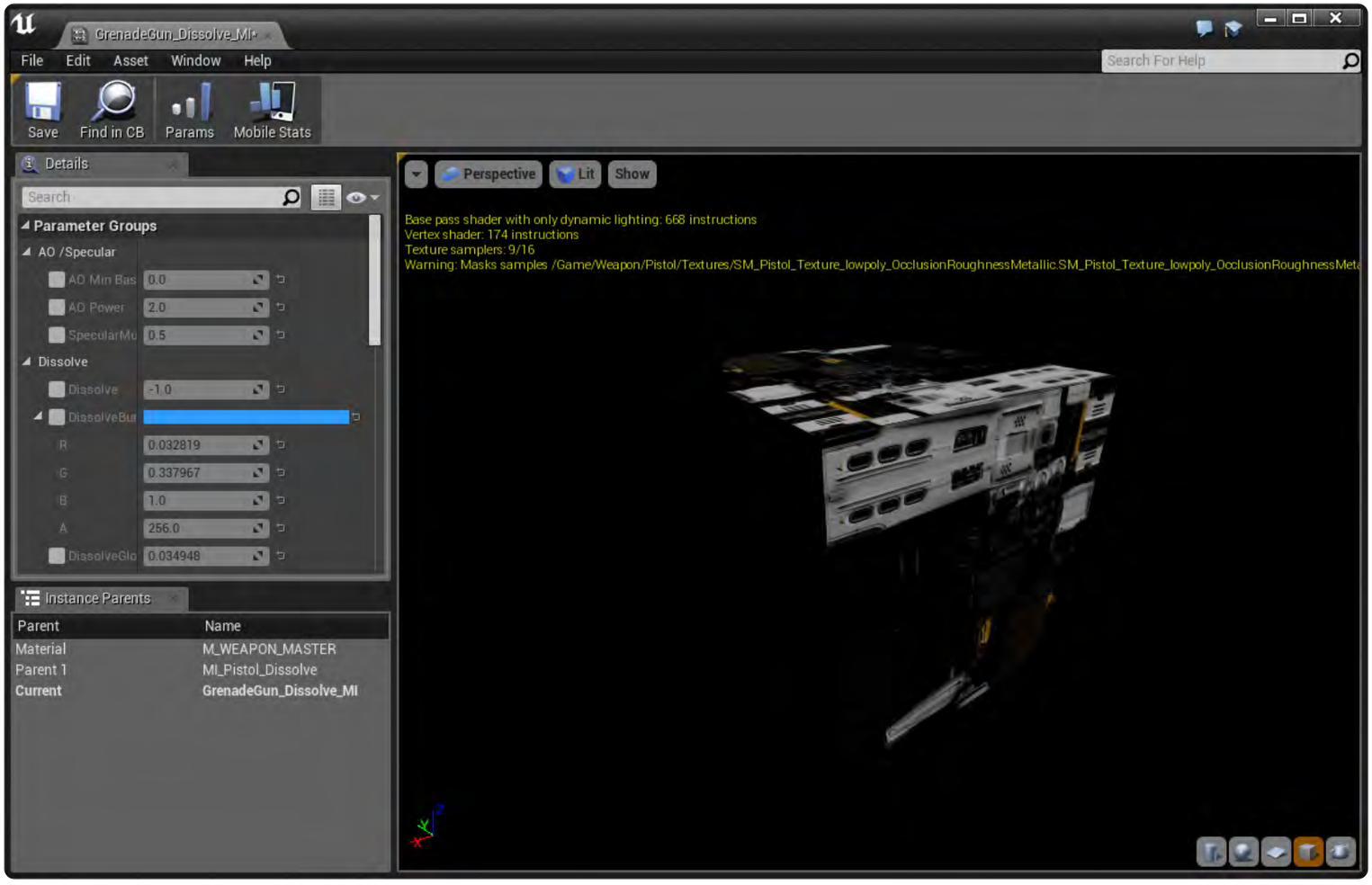
3. Drag the new Material Instance to the **GrenadeGun Content** folder. Choose **Move Here** when you release the mouse button to move it to that location.



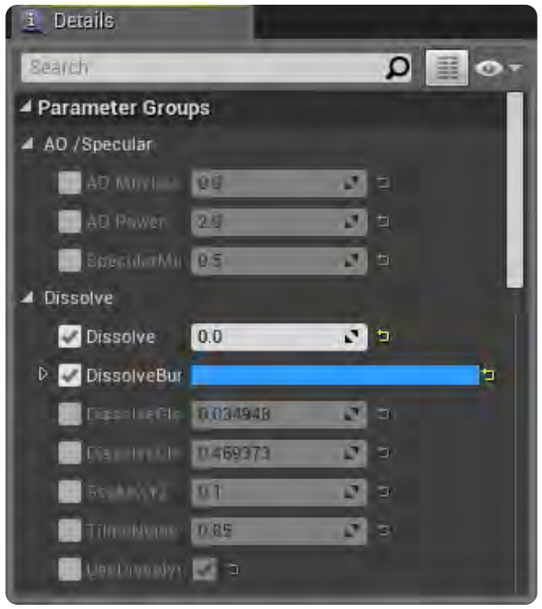
4. Press **F2** to rename the Material Instance and enter **GrenadeGun_Dissolve_MI** as the new name.



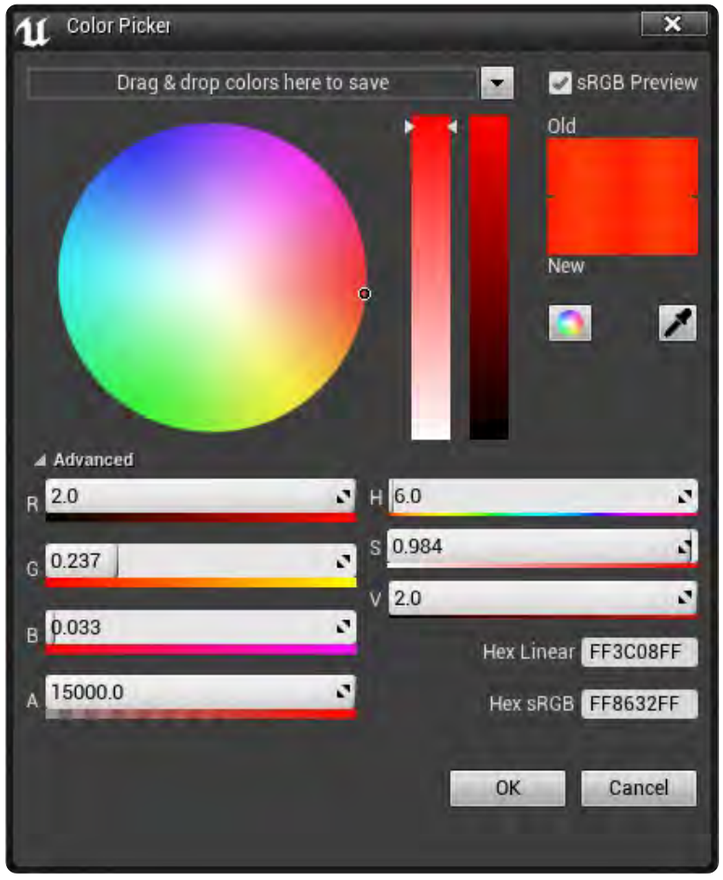
5. **Double-Click** the *GrenadeGun_Dissolve_MI* to edit it in the Material Instance Editor .



6. Click the checkbox next to the **Dissolve** parameter to enable overriding it. Set the value of the **Dissolve** parameter to a value in the range of `-0.2` to `0.33` - the value you use will depend on the preview shape you are currently using in the Preview viewport - to cause the dissolve effect to be visible.
7. Click the checkbox next to the **DissolveBurnColor** parameter to enable overriding it. This parameter controls the color used for the dissolve effect.



8. Click the colored bar to summon the Color Picker and set it to a red-ish color. Here we've used the values of: (2.0, 0.23705, 0.0032819, 15000.0)

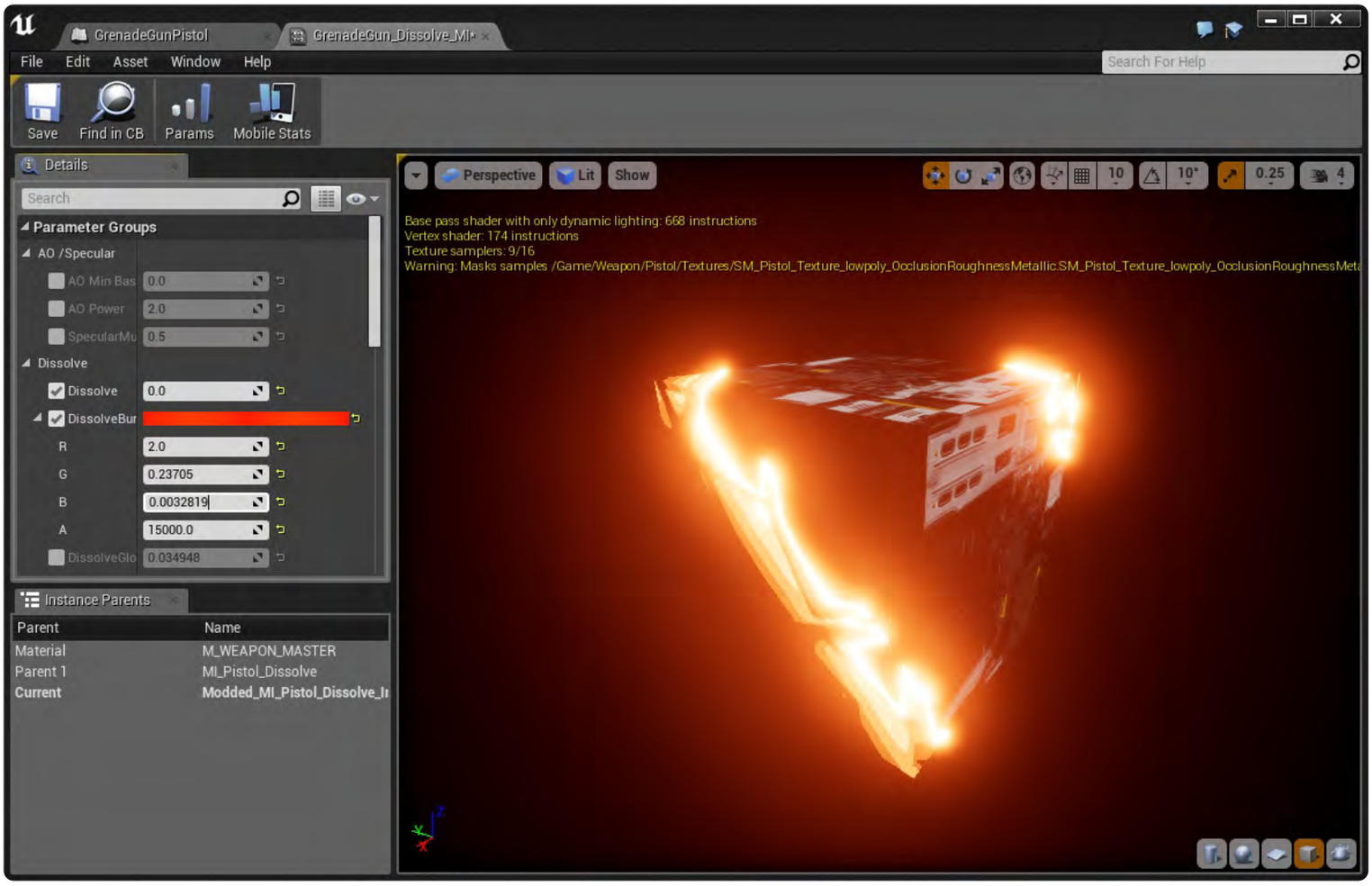


You can also enter the R, G, and B values directly in the Details panel by expanding the **DissolveBurnColor** parameter to expose the individual values.

9. In the Material Instance Editor toolbar, Click the checkbox next to the **Dissolve** parameter again to disable it since we were just using that for preview purposes.
10. Click the **Save** button to preserve your changes

Results

You now have a version of the dissolve Material with a red effect instead of the default blue effect.



Previous Step

Overriding and Extending Functionality

Next Step

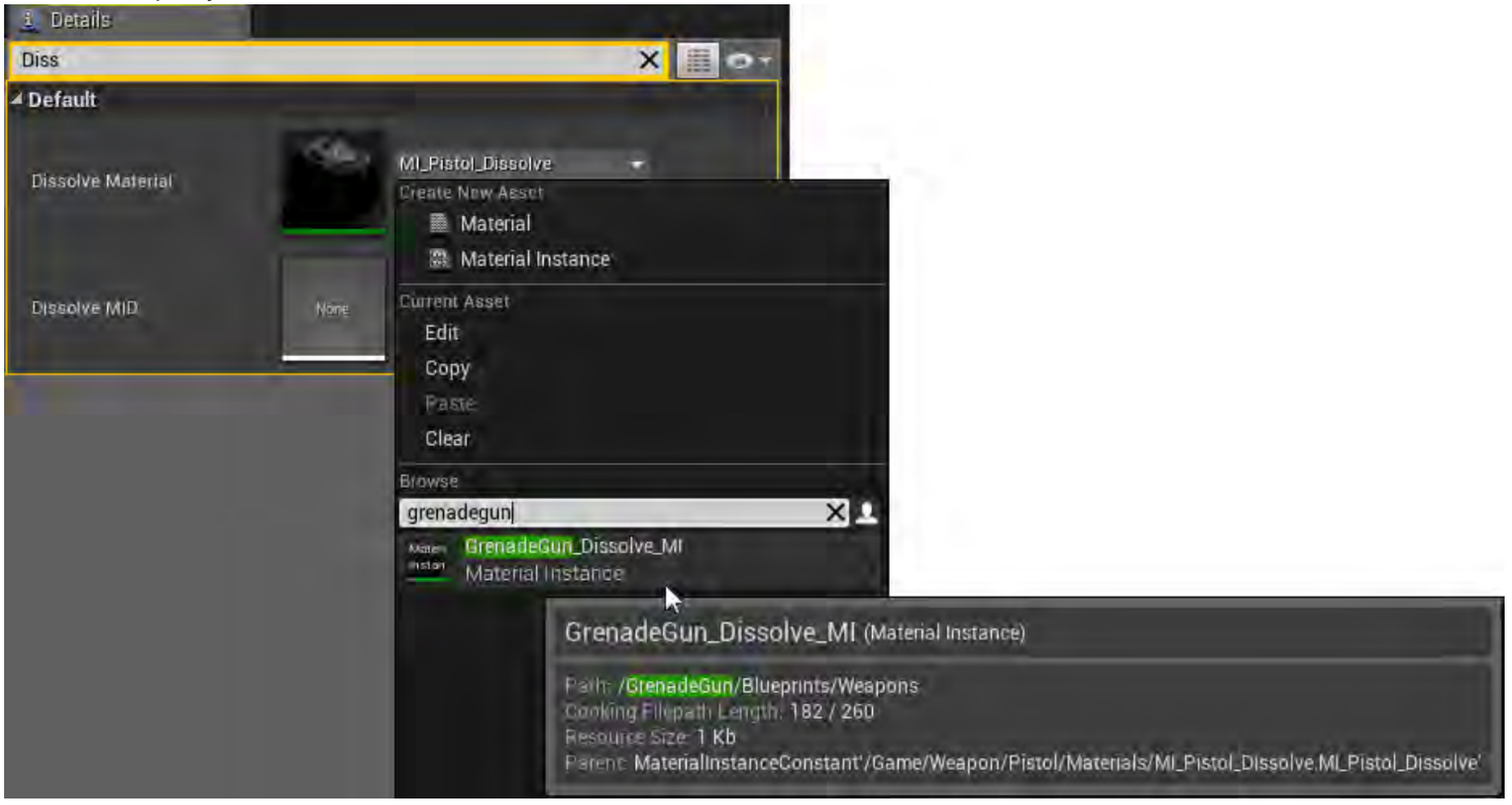
3. Assigning the Material Instance

Your custom pistol needs to know about the new dissolve Material in order to make use of it. You can assign it to your custom pistol to force it to be used.

Steps

1. Double Click the **GrenadeGunPistol** Blueprint to open it in the Blueprint Editor .
2. Select **GrenadeGunPistol(self)** in the **Components** panel

3. In the Details panel search for **Dissolve** to easily find the **Dissolve Material** property.
4. Click the dropdown menu for the **Dissolve Material** property to show the Asset Picker and select **GrenadeGun_Dissolve_MI**. You can use the search box to quickly filter the list.



5. In the Blueprint Editor toolbar, click **Compile** to update the Blueprint and click **Save** to preserve your changes.

Results

That's it, the GrenadeGunPistol, when it is destroyed, will now dissolve to red instead of blue.

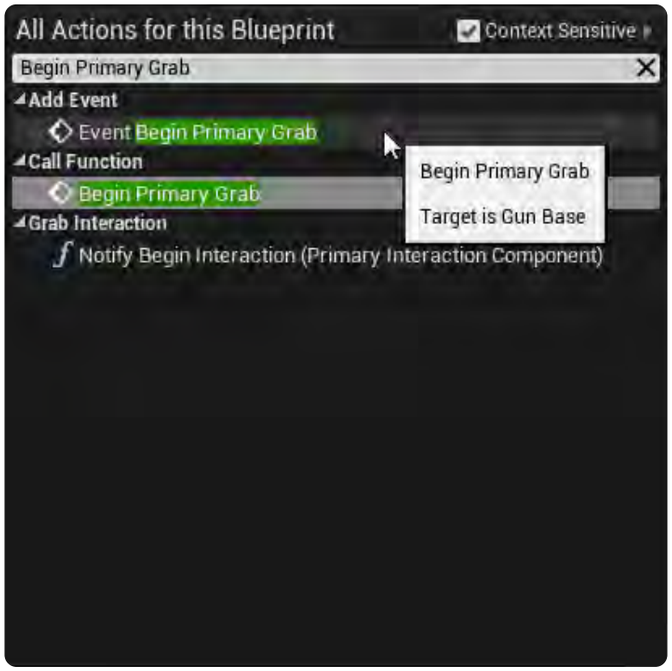
Previous Step | Overriding and Extending Functionality | Next Step

4. Building the Grab Events

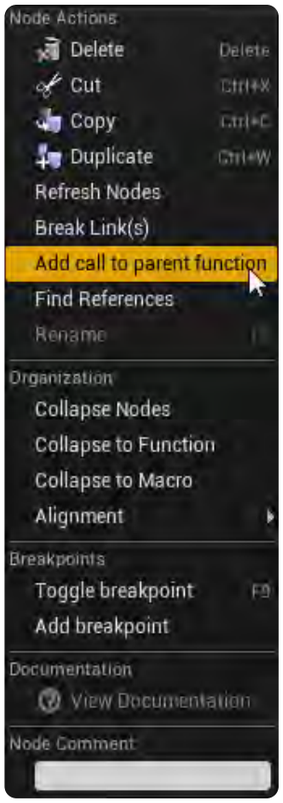
In this step you will be overriding the **Grab Events** on our GrenadeGunPistol so we can check to see if we want the gun to explode or not when it collides with things.

Steps

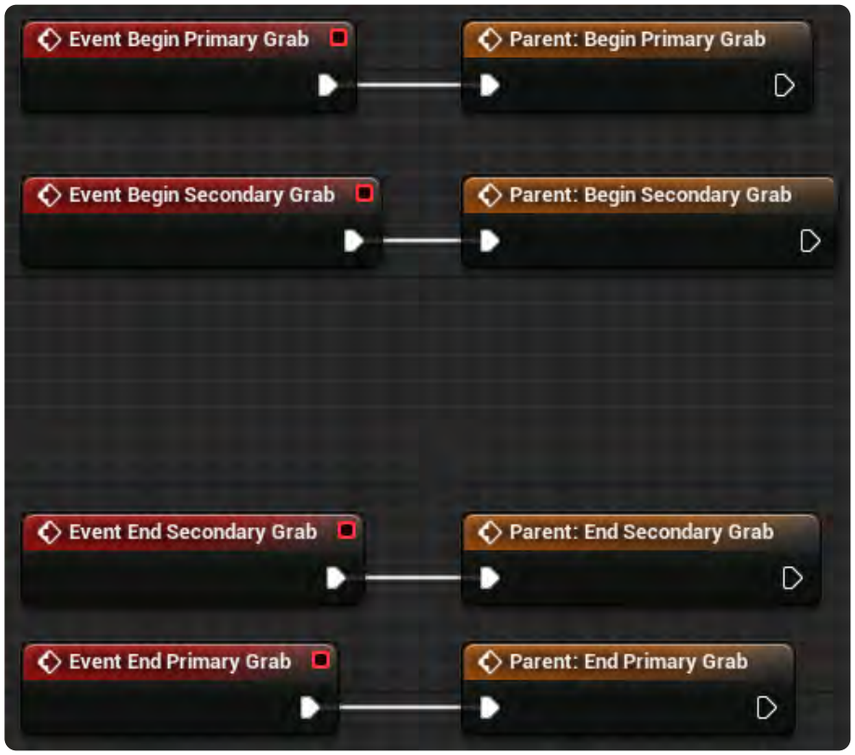
1. **Right-click** in the Event Graph and search for **Begin Primary Grab**. Select **Event Begin Primary Grab** to add the node to the Event GRaph.



2. Repeat this process for:
    - **Event Begin Secondary Grab**
    - **Event End Primary Grab**
    - **Event End Secondary Grab**
3. For each of these Event nodes, you need to make a call to the same Event in the parent Blueprint to make sure that any existing functionality is still executed. To do this, **Right-click** the four event nodes you just created and select **Add call to parent function**.

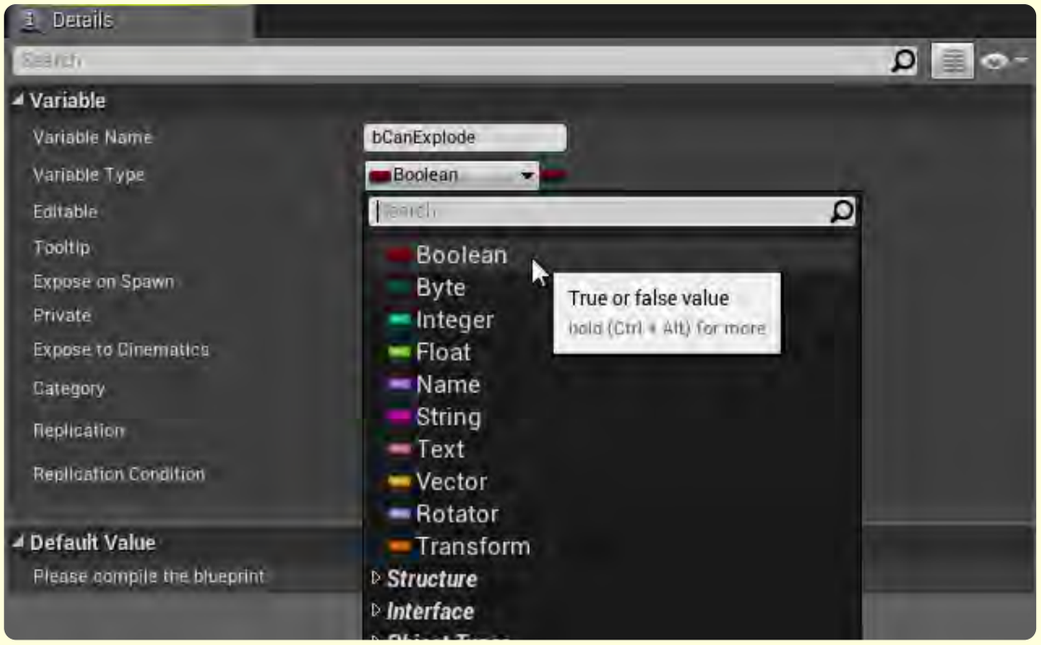


4. Connect the Event Nodes to their Parent Nodes.

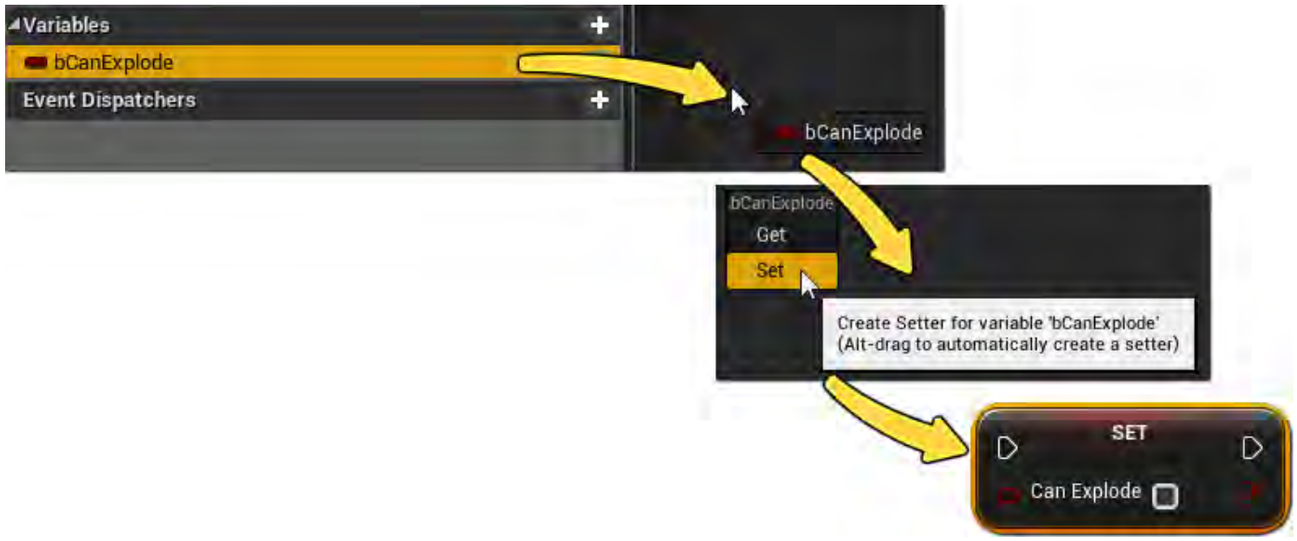


5. Click the **+** button next to **Variables** in the **My Blueprint** panel to create a new variable and name it **bCanExplode**.

If you haven't created a new variable in this session in Unreal Editor, it will automatically be a **Bool** (the red icon indicates that it is a **Bool**). However, if you need to change it, select the variable in the **My Blueprint** tab and then you can change it in the **Details** panel:

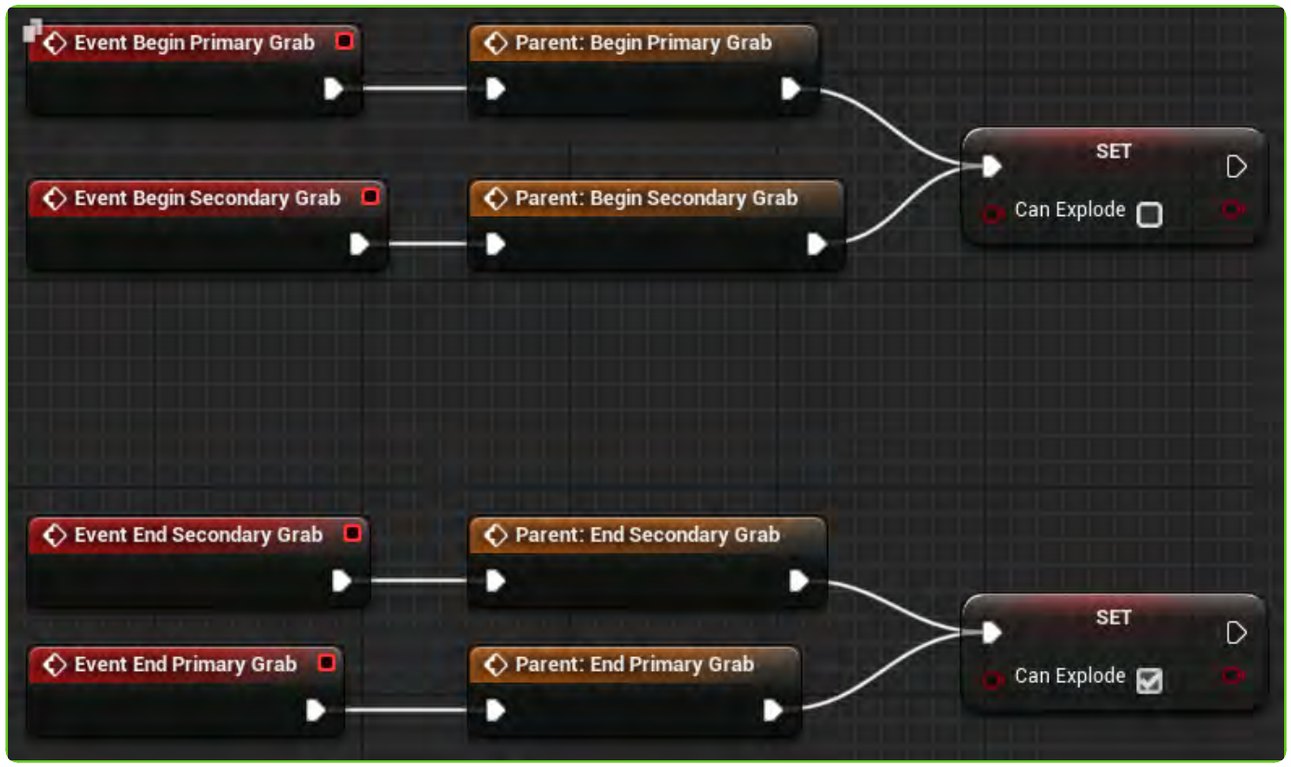


6. **Click + Drag** the **bCanExplode** variable into the **Event Graph**, and select the **Set** option.



7. Duplicate the **Set Can Explode** node using **Ctrl + W** and click the checkbox in the new node to set the value of **bCanExplode** to **True**.

8. Finally, connect the Parent nodes to the corresponding **Set Can Explode** nodes as shown below.



Click the icon in the upper left corner of this image to copy the Blueprint Graph and paste it into your project.

9. In the Blueprint Editor toolbar, click **Compile** to update the Blueprint and click **Save** to preserve your changes.

Results

These sets of Events are called when our gun is grabbed and released. You'll use the **bCanExplode** variable in the next section of the Blueprint to prevent the gun from exploding in your hand when you punch a robot with it.

Previous Step | Overriding and Extending Functionality | Next Step

5. Creating the Base Damage Blueprint

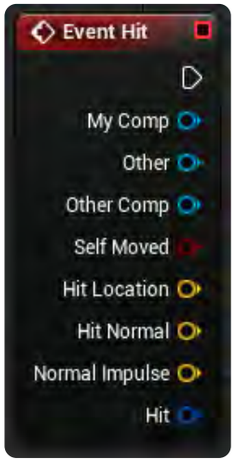
Previous Step | Overriding and Extending Functionality | Next Step

This step will cover building the basic Blueprint for having the GrenadeGunPistol deal damage once it collides with the something.

Since you've learned how to make nodes in the previous steps, this section will start to accelerate by grouping creating nodes into chunks. Just remember you can **Right-click** in the **Event Graph** to find every node available to the Blueprint, and pulling off a pin will narrow down the context sensitivity of the node search to things that operate on that pin type.

Steps

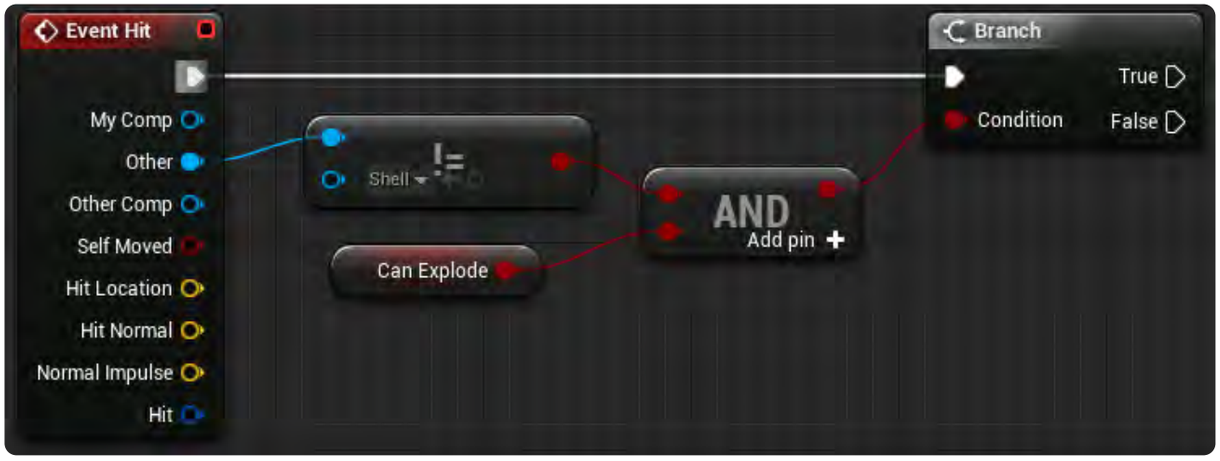
1. Create an **Event Hit** node.



2. Pull off the **Other** pin and search for **!=** to create a **Not Equals** node.

3. Set the second input pin on the **Not Equals** node to `shell`, you'll need to scroll down a ways to find the Blueprint simply called `shell`.

4. Next up you'll need to setup a bit of logic using an **And** node, a **Branch** node, and the **bCanExplode** variable.

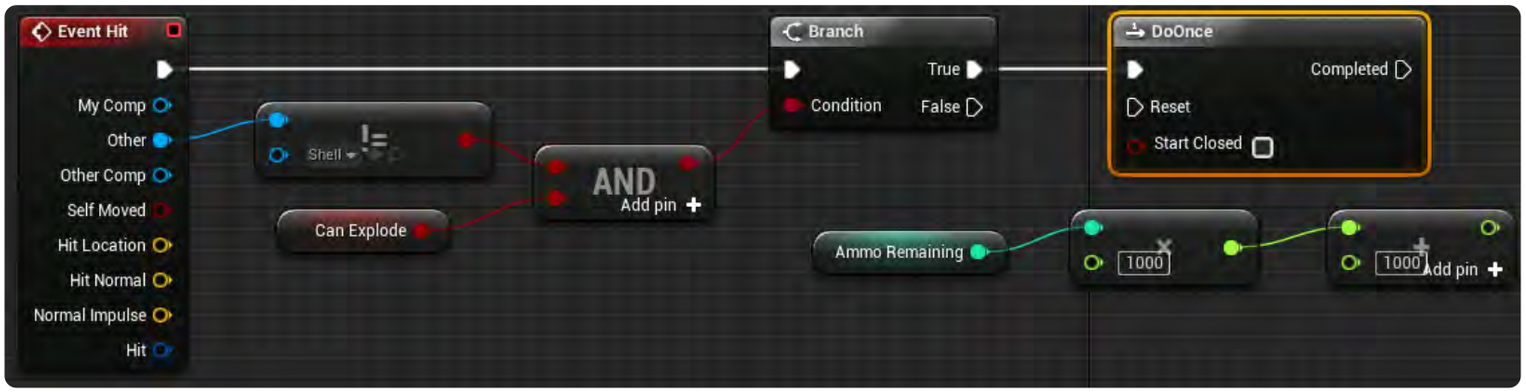


> There are a number of keyboard shortcuts in the **Event Graph**, once of which is **Hold B + Left Click** to create a **Branch** node, arguably one of the most used logic nodes in **Blueprints**.

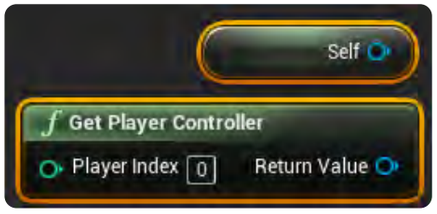
5. Now you're going to set up how much damage the GrenadeGun will do by getting the **Ammo Remaining** in the gun, multiplying it by 1000 and adding 1000 to it.

Ammo Remaining is a variable that is apart of the base pistol class, so you can find it by searching for **Get Ammo Remaining**. The **Multiply** node is actually an **Int * Float** node, and can be found by searching for **Int * Float**. And finally, the **Addition** node can be found by searching for **Float + Float**.

6. You'll only want the gun to explode once, so you'll need a **Do Once** node connected to the **True** pin of the **Branch** node.



7. You're going to need a **Self** node and a **Get Player Controller** node.



8. Finally, to do some damage you'll need a **Apply Radial Damage** node, that will take a bunch of inputs from what you've created and will require some settings changed. Connect it up as shown below. You'll need to change 3 settings on **Apply Radial Damage** node: **Damage Radius**, **Damage Type Class**, and **Do Full Damage**.
    - Set **Damage Radius** to `400`
    - Set **Damage Type Class** to `Odin Damage Type`
    - Set **Do Full Damage** to `True`**

Click the icon in the upper left corner of this image to copy the Blueprint Graph and paste it into your project.

> You can create **Reroute Nodes** to clean up your graphs by **Double Clicking** on any connecting line. As seen above on the yellow line that leads from **Hit Location** to **Origin**.

9. In the Blueprint Editor toolbar, click **Compile** to update the Blueprint and click **Save** to preserve your changes.

Results

After this step, you will have a functional weapon that will explode when thrown into the world. It will do more damage based on the ammo left in the magazine, and a fully loaded pistol will do enough damage to kill just about any non-specialized bot. In the next step, you will be adding in sound, VFX, a physics impulse, and clean up the gun after it explodes.

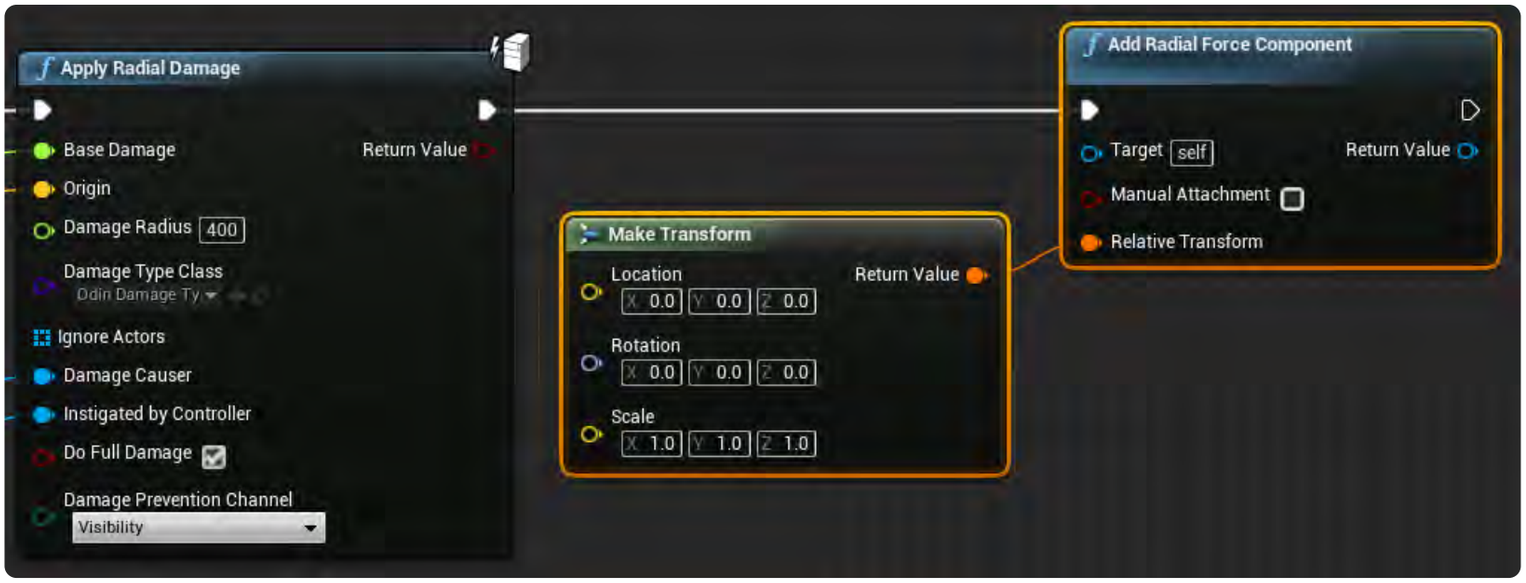
6. Adding Visual Effects

In this step you will be finishing off the GrenadeGunPistol Blueprint by adding a particle system, sound, a radial force, and finally cleaning up the gun after the explosion.

Steps

1. Using the techniques learned in previous sections, create a **Add Radial Force Component** node.

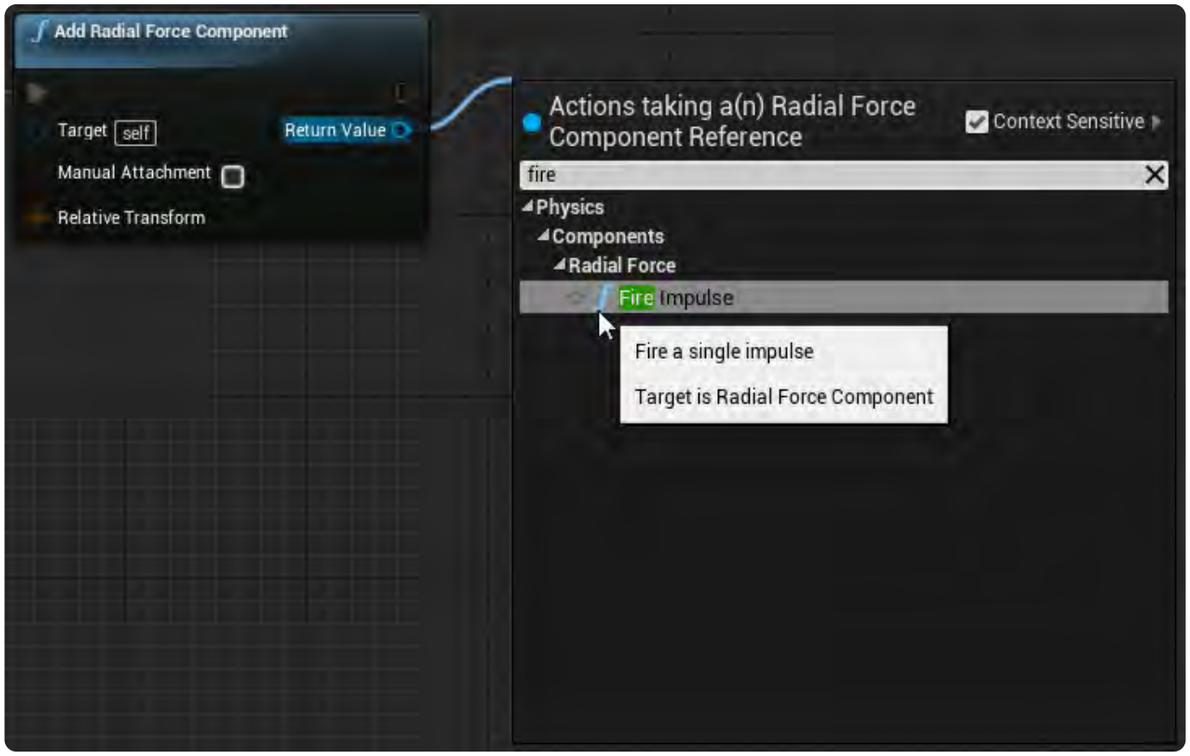
2. It needs a transform, so create a **Make Transform** node and connect its output to the **Relative Transform** pin on the **Add Radial Force Component**.



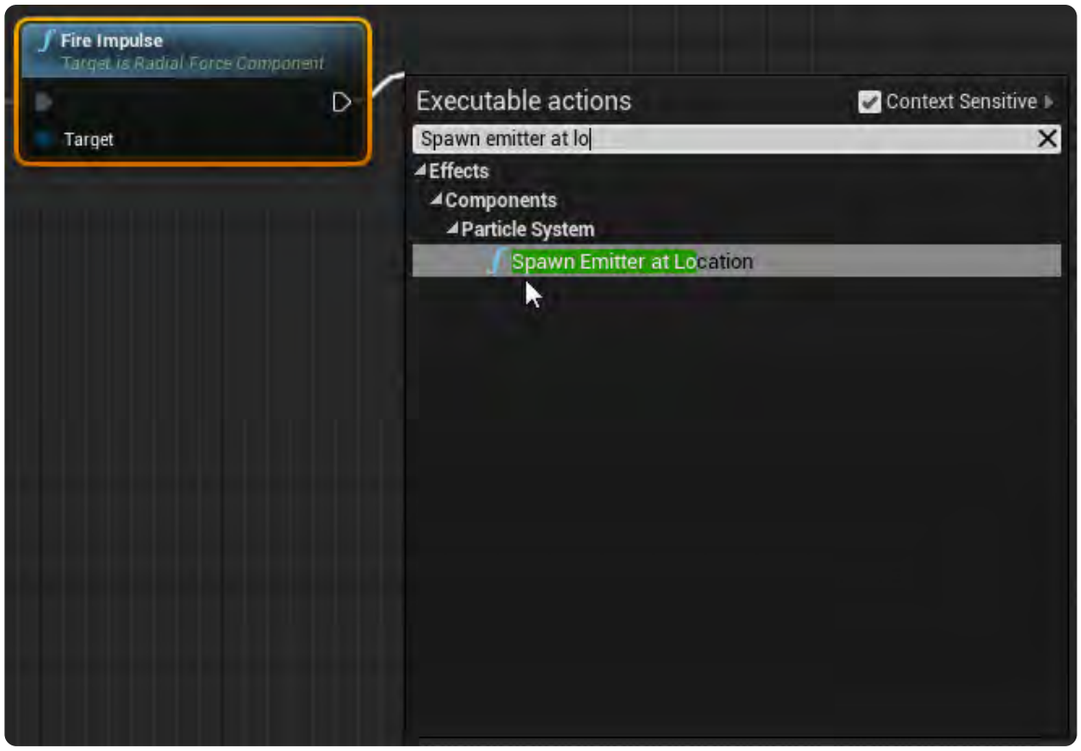
3. Select the **Add Radial Force Component** and change **Impulse Strength** to `650.0`. Enable **Impulse Vel Change**, Enable **Ignore Owning Actor**, and **Force Strength** to `0.0`.

4. Also, using the **+** sign on the **Object Types to Affect**, add a new entry and change it to **WorldDynamic**. This will allow the force to hit the bot ragdolls.

5. Next, pull off the **Return Value** on the **Add Radial Force Component**, search for and create a **Fire Impulse** node.

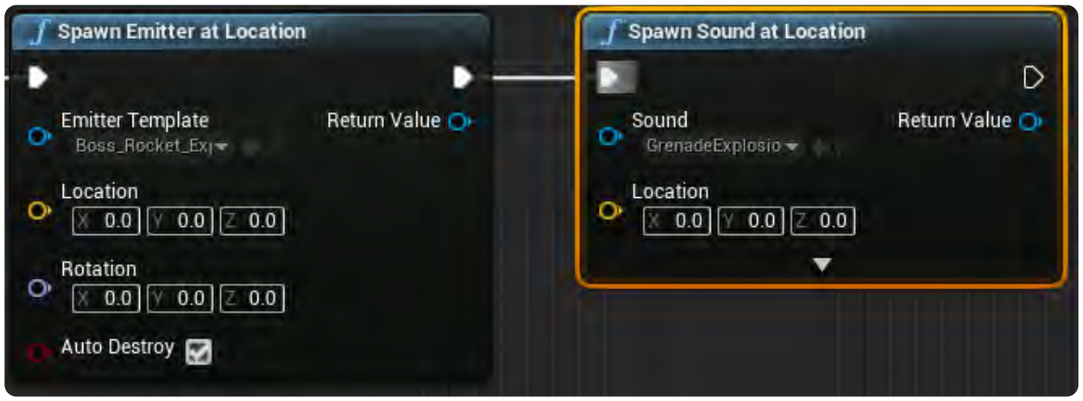


6. From the output execution pin of that node, add a **Spawn Emitter at Location** node.



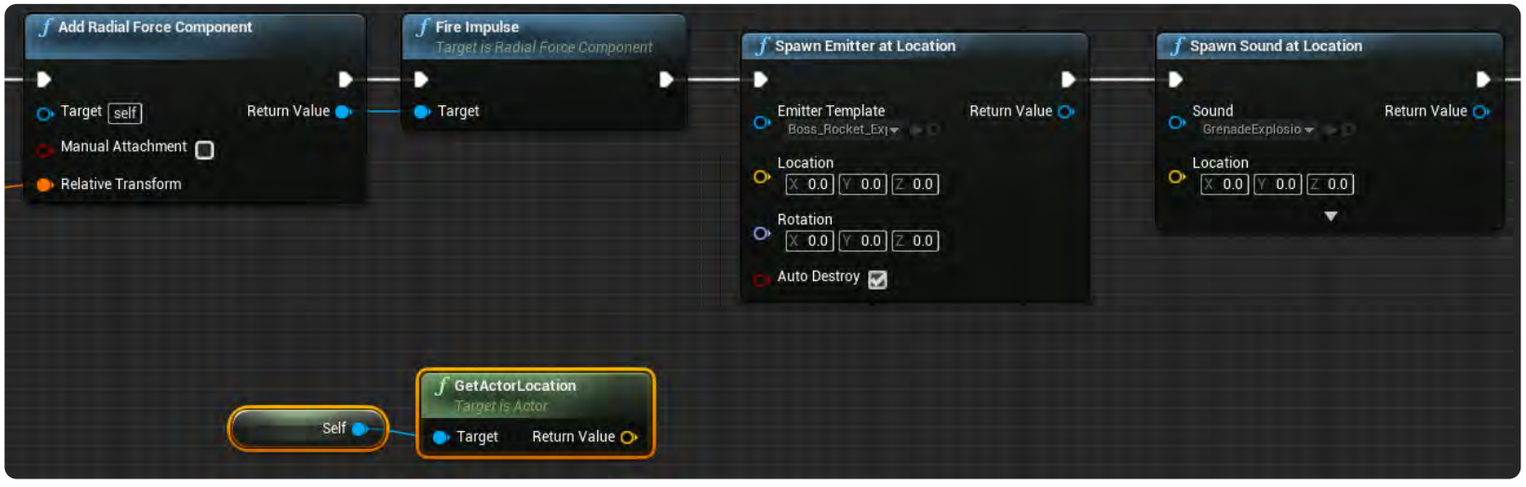
7. Set its **Emitter Template** to `Boss_Rocket_Explo`.

8. From the output execution pin of that node, add a **Spawn Sound at Location** node.
9. Set its **Sound** to `GrenadeExplosion_C_Cue`

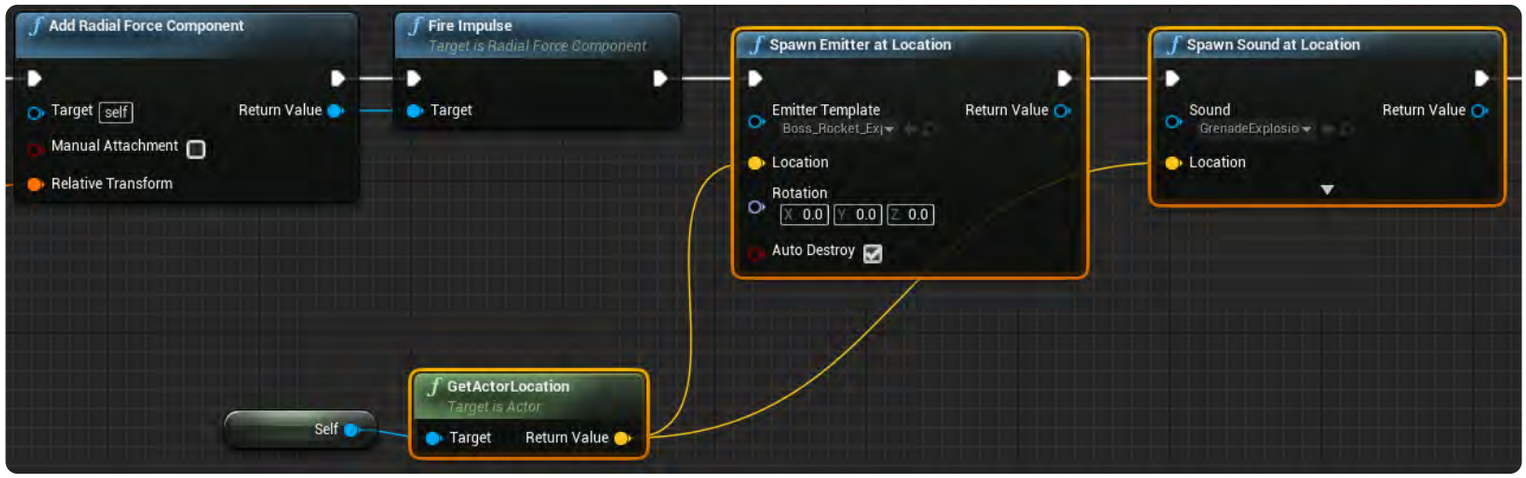


10. From the output execution pin of that node, add a **Set Ammo Remaining** node, which should come in set to 0.
11. From the output execution pin of that node, add a **Force Cleanup** node, which will dissolve and destroy the gun.

12. Create are a **Self** node (search for **Get a reference to self**). Pull off **Return Value** and create a a **Get Actor Location** node.

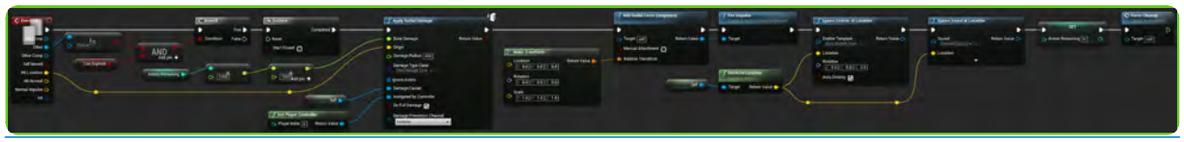


13. Connect the output of the **Get Actor Location** node to both the **Spawn Emitter** and **Spawn Sound at Location** nodes.



14. Below you can see the full graph.

> The graph below has been cleaned up using **re-route nodes**. These are essentially pass-through nodes that enable you to better organize the connections in a graph and make the graph easier to read.



Click the icon in the upper left corner of this image to copy the Blueprint Graph and paste it into your project.

Results

The GrenadeGunPistol is now complete. In the final step, you'll go over how to test the gun and ideas on where to take it next.

‹ Previous Step | Overriding and Extending Functionality | Next Step ›

7. Testing and Next Steps

With your custom pistol completed, you can click **Play** while the the hub level is loaded, and enable your GrenadeGunPistol through the mod menu. You can now use the Gun Range to toss the gun at the targets and watch them detonate!

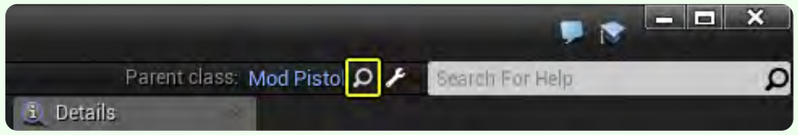
Next Steps

Now that you successfully created a new pistol, it's time to see if you can take it further! If you're having a creative block, here are some ideas to get the creativity flowing while also getting you to dive further into the gun Blueprints.

- Balance the damage of the explosion
- Make the explosion radius dependent on how many bullets are left in the gun
- Have the gun bounce once or twice before exploding
- By disconnecting the **Instigated by Controller** pin in the GrenadeGunPistol Blueprint, you can have the explosion damage the Player
- Why not more explosions? Make it a MIRV Grenade!
    - **Bonus:** Going with a more literal definition of MIRV? Have it call down a ballistic missile strike from space!
- Make the pistol a Machine Pistol with extended magazines and fully automatic firing!
    - **Bonus:** Adjust the damage that the Machine Pistol does to be more balanced for its new fire rate
    - **Mega Bonus:** Make the bullets it fires homing rounds!

- If you plan to continue using the base mesh for the pistol, figure out how to have the Blueprint add Attachments (laser, compensator, sights, etc…)
- Give the gun bonus damage on its next shot based on how many times you juggle it

In the top right of the GrenadeGunPistol Blueprint you will see a little magnifying glass:



This button will take highlight the current Blueprint's Parent in the Content Browser. By using this button you can navigate down to Gun_Pistol and Gun_Base, which contain a mountain of functionality for the Pistol. Browse through them to see what variables and functions you have available to you.

Previous Step | Overriding and Extending Functionality | Next Step

Adding Gameplay To Your Map

Steps

1. 1. Creating a Map Mod
2. 2. Placing Gameplay Actors
3. 3. Setting Up Waves of Robots
4. 4. Level Startup Scripting
5. 5. Level Completed Scripting

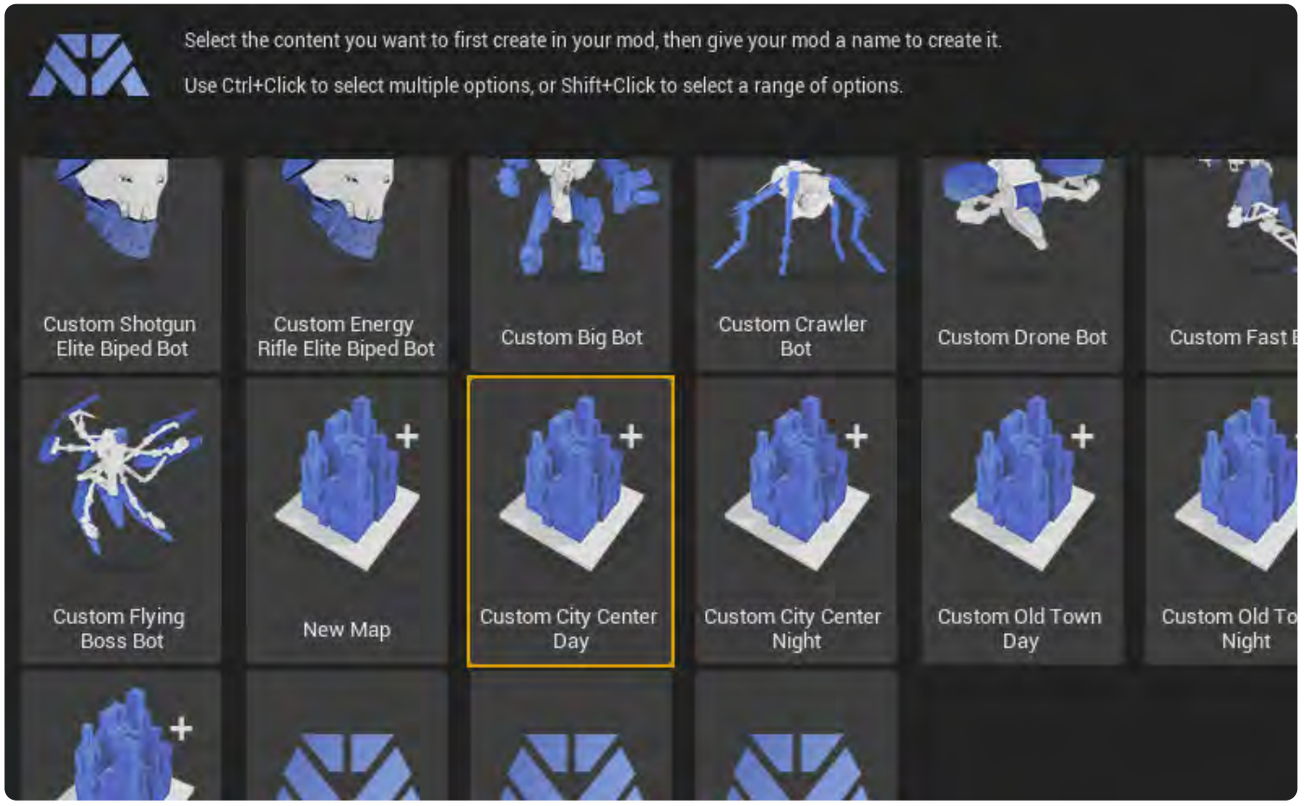
Click to Start

1. Creating a Map Mod

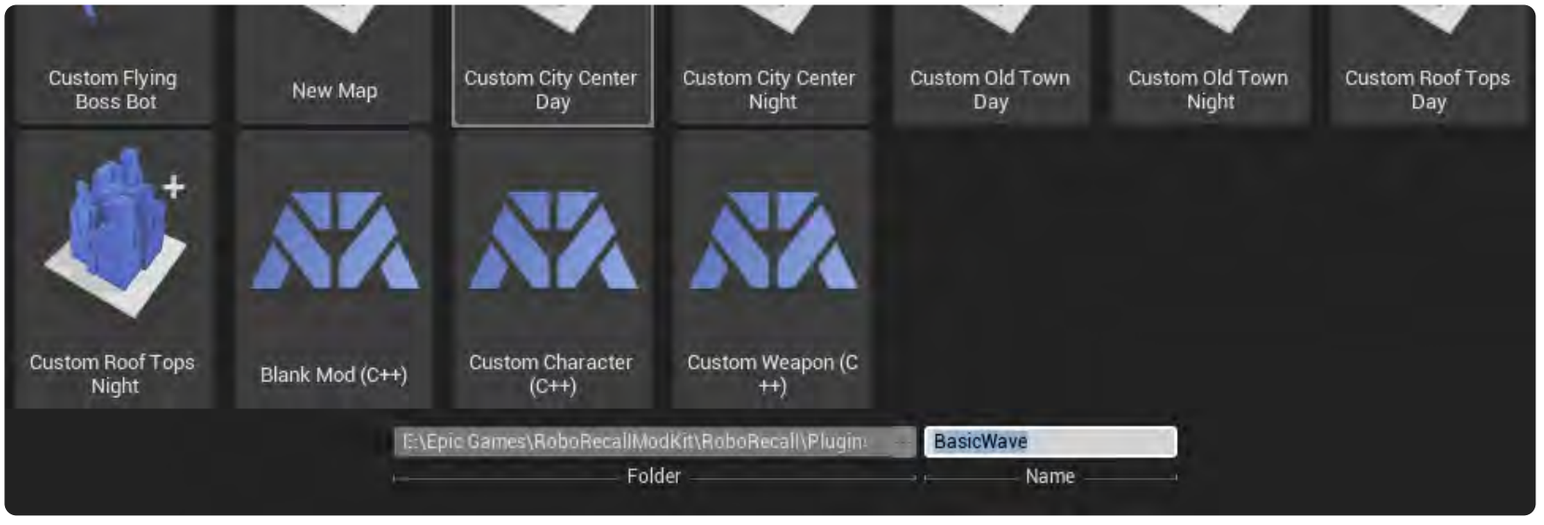
Previous Step | Adding Gameplay To Your Map | Next Step ›

This instructions below walk you through creating a new map mod from scratch. If you already have a map created, you can skip this step and move on to placing Actors .

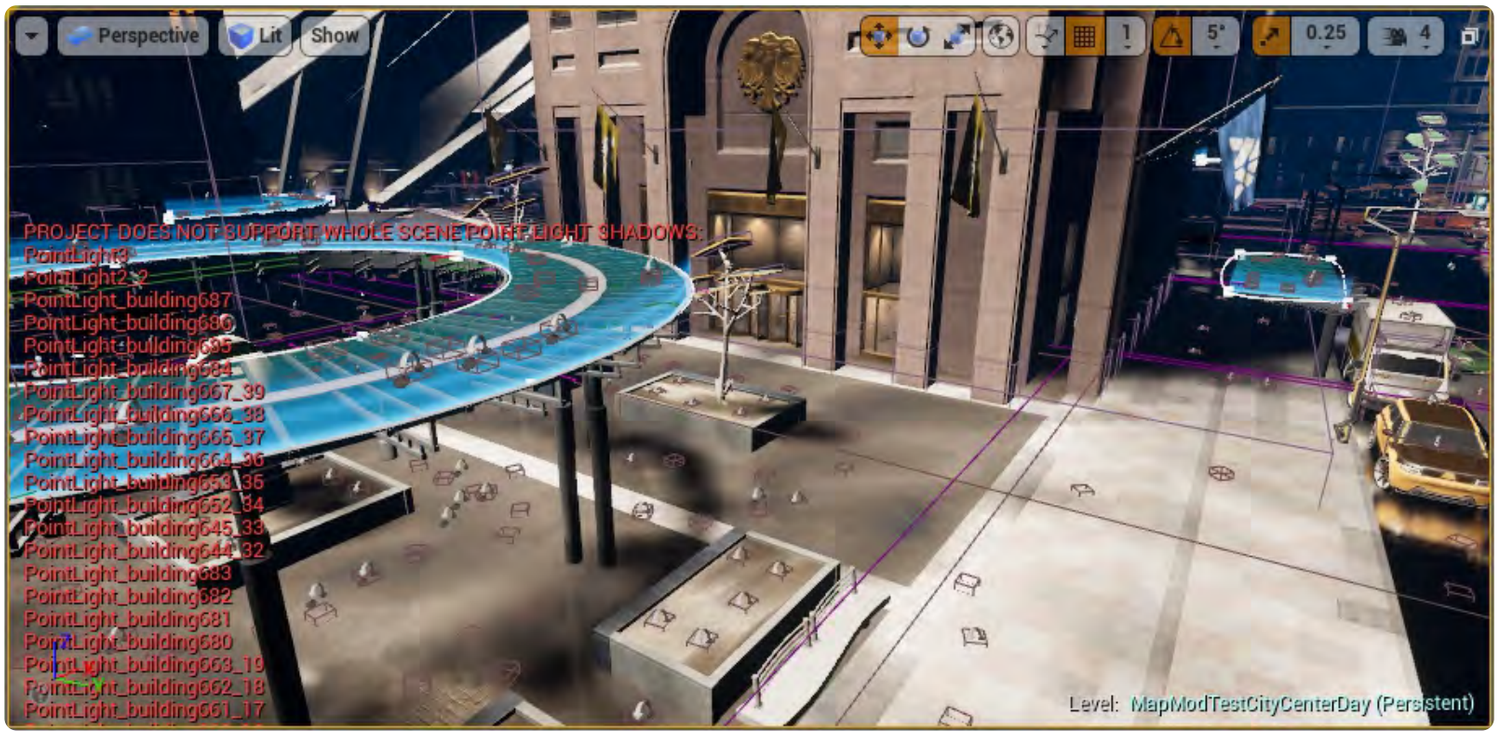
1. In the Level Editor toolbar, click the **Create Mod** button to open the **New Game Mod** window.
2. Select one of the map templates. In this example, we chose **Custom City Center Day**.



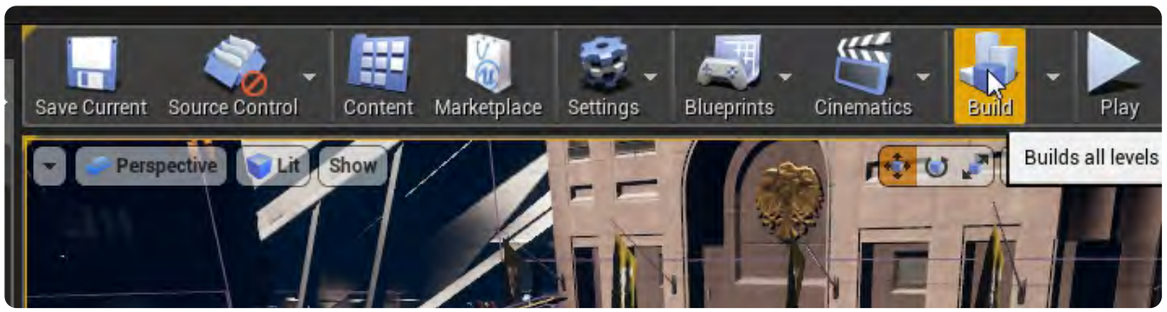
3. Enter a name for the new mod in the **Name** field.



4. Click the **Create Mod** button to generate your mod files.
5. In the **Content Browser**, double-click the **Maps** folder and then double-click your map asset to load your new map.



6. In the Level Editor toolbar, click the **Build** button to generate lighting, navigation, and other data for your map.



Result

You now have a map that provides a perfect environment to add gameplay elements to.

Previous Step

Adding Gameplay To Your Map

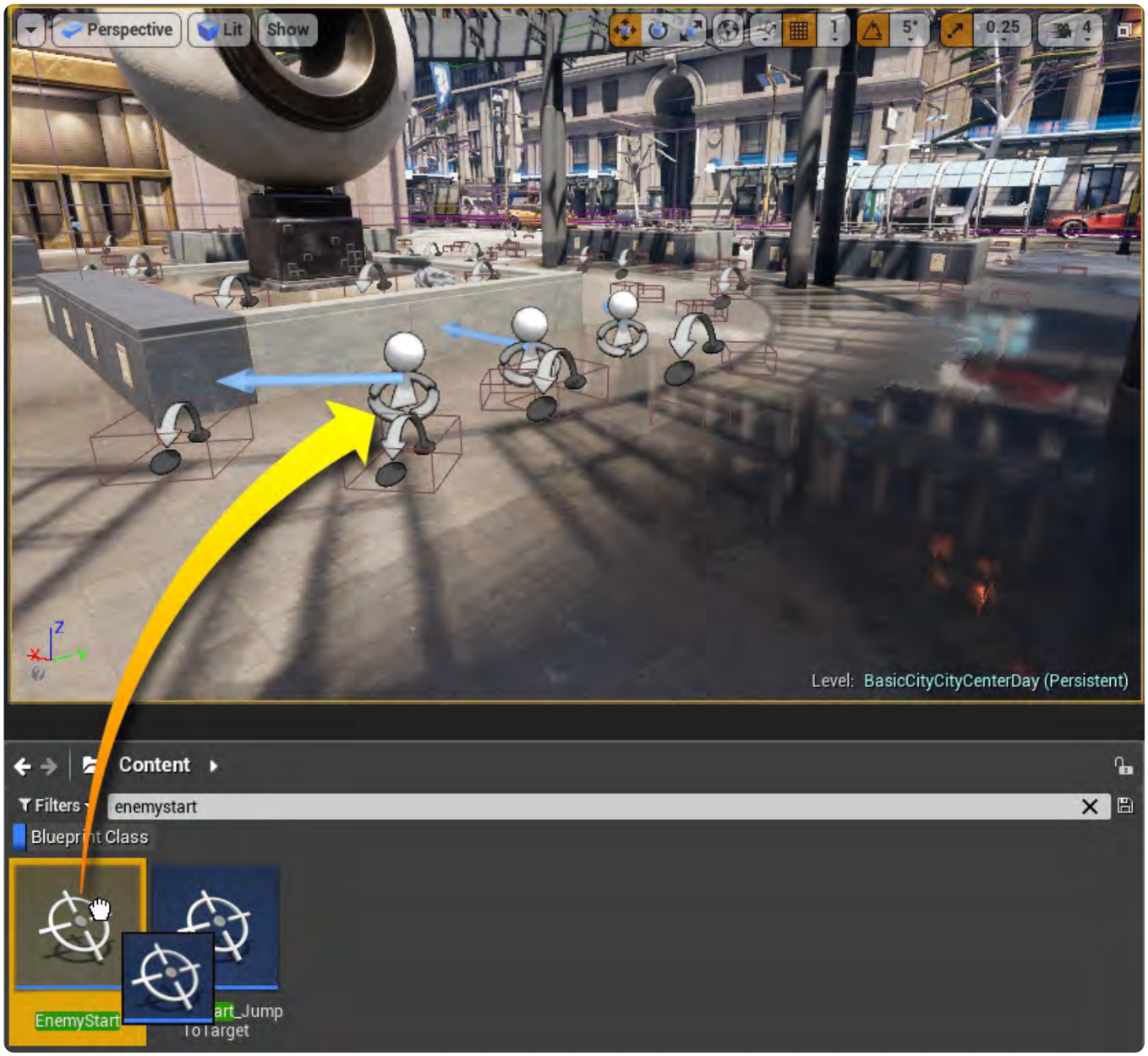
Next Step

2. Placing Gameplay Actors

1. In the Modes panel , select the **Place** mode.

2. Select the **Basic** tab and drag a **Player Start** Actor into the viewport. This will act as the starting location for the player, so place it where you want the player to begin the level from.

3. In the **Content Browser**, select the **Content** folder and search for **EnemyStart**. Drag a few instances of this Blueprint into the viewport. These act as locations to spawn the robots from. Place them wehre you want the robot waves to eminate from.



4. In the **Content Browser**, select the **C++ Classes** folder and search for **OdinAIWaveManager**. Drag an instance of this Actor into the viewport. The location is not important as its job is just to act as a controller for spawning the robots, but you should place it somewhere easy to find and select.

Result

You now have all the Actors necessary to create a wave of attacking robots. There are some properties to set and connections to make before they will do anything, so let's do that now!

Previous Step
Adding Gameplay To Your Map
Next Step

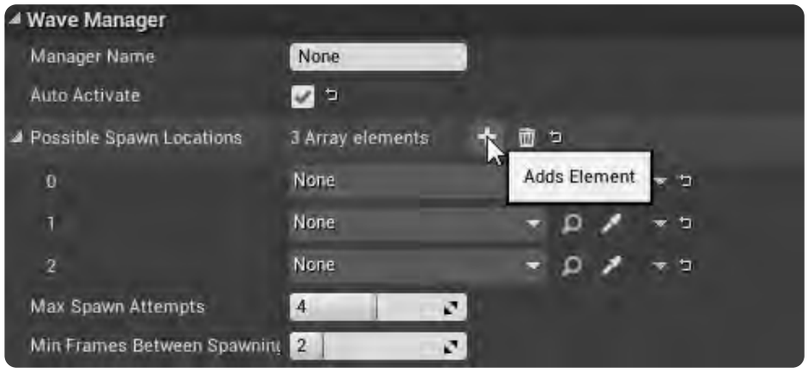
3. Setting Up Waves of Robots

The Wave Manager controls the generation of waves of attackling robots. You can tell it what types of robots to spawn, where to spawn them from, how many to spawn, etc. Once you set up the properties for the Wave Manager, robots will begin to spawn and attack just like in the game.

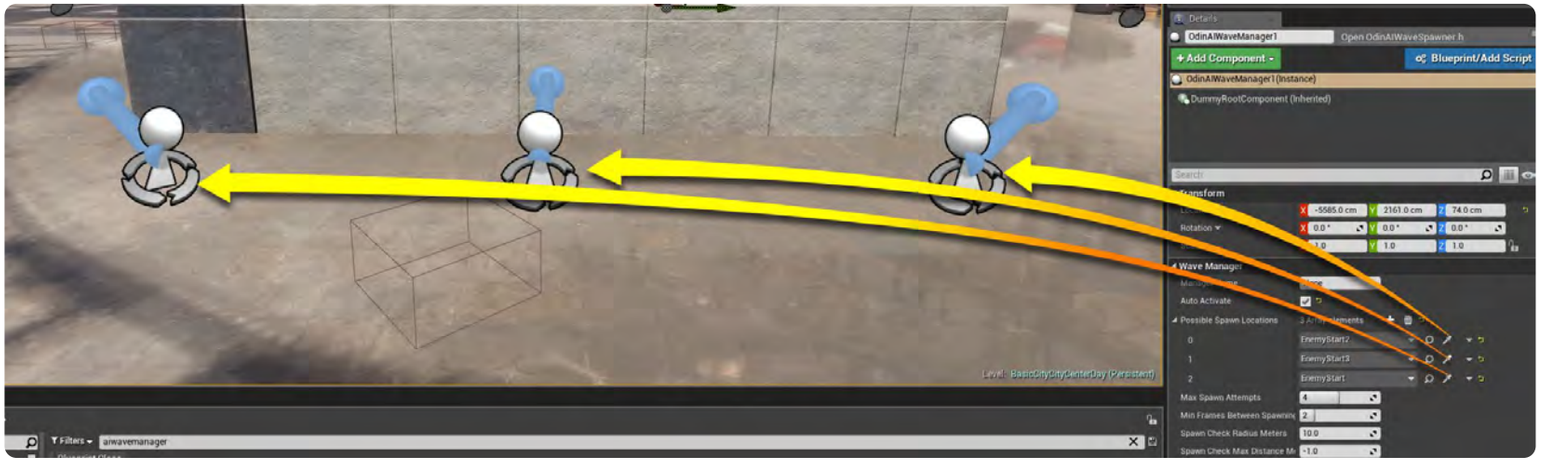
1. Select the **OdinAIWaveManager** Actor you previously placed to view its proeprties in the **Details** panel.
2. Enable the **Auto Activate** checkbox to force the waves to automatically begin spawning when the level begins.

   > You can also leave this disabled and enable it during play in response to some event in the game using the Level Blueprint if that better fits the gameplay you are trying to create.

3. The **Possible Spawn Locations** property determines where the robots will spawn from for the wave. Click the **+** button to add elements to the array. You need one for each **Enemy Start** Actor you placed previously.

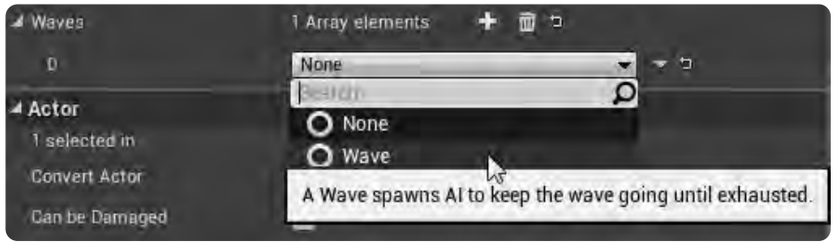


4. Click the **Pick Actor from Scene** button for each element in the array and then select one of the **Enemy Start** Actors in the viewport to assign it to the element.

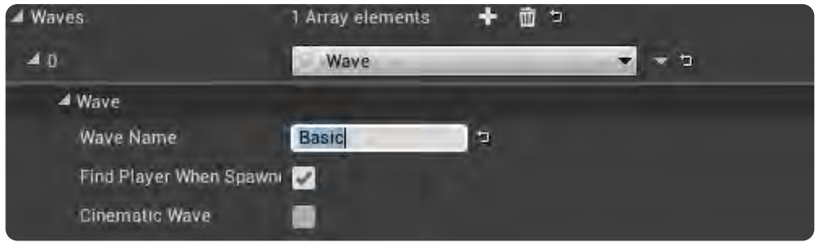


5. The Wave Manager can control any number of waves of robots. You specify the waves of robots by adding elements to the array. CLick the **+** button to add a new wave.

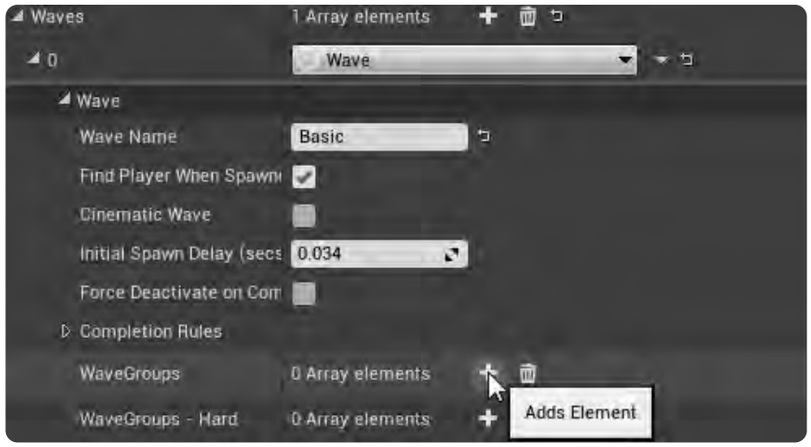
6. You must set the type of wave to see the available properties. Fortunately, there is only one type available currently: Wave. Choose **Wave** from the class picker.



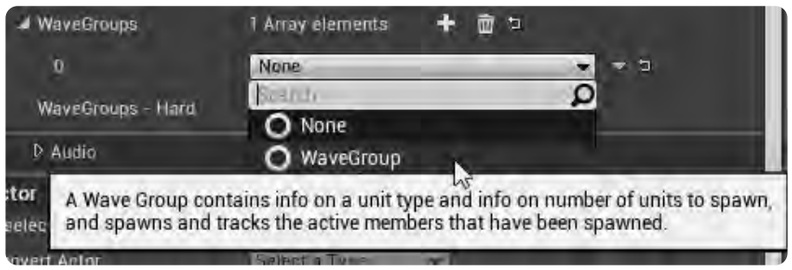
7. Enter a name for the wave using the **Wave Name** property. When dealing with multiple waves, giving each wave a name enables you to set up wave-specific behavior in the Level Blueprint.



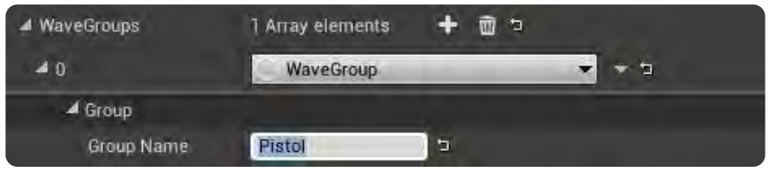
8. Each wave can have any number of groups, which are collections of robots of the same type. Click the **+** button for the **WaveGroups** property to add a new wave group.



9. As with the wave, you need to set the type of the group, and again there is only one type available: *WaveGroup*. Choose **WaveGroup** from the class picker.

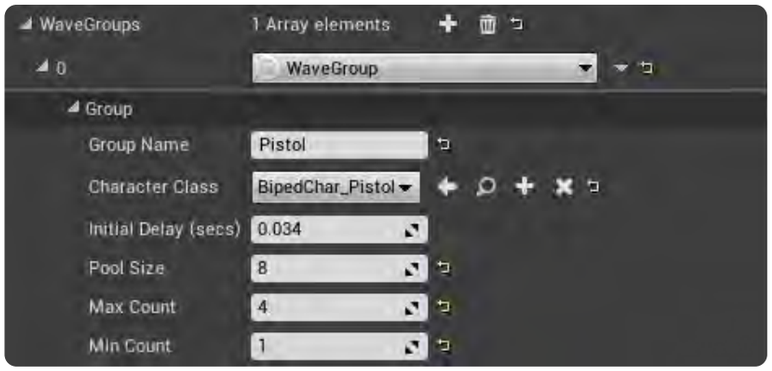


10. Give the wave group a name using the **Group Name** property. Again, this makes it possible to set up scripted events related to a particular group.



11. The **Character Class* property determines what type of robots the group contains. There are a variety of types available, but for this example we will use a basic pistol-carrying robot. Choose** BipedChar_Pistol** from the class picker.

12. Finally, you need to specify how many robots the group will contain total, how many to spawn at once, and when to spawn them. **Pool Size** is the total number of robots the group will spawn. **Max Count** is the number that are spawned at once. **Min Count** determines when to spawn new robots. The Wave Manager will spawn new robots when there are only **Min Count** left. Set **Pool Size**, **Max Count**, and **Min Count** to 8, 4, and 1 respectively. Using these values, the Wave Manager will initially spawn 4 (Max Count) robots. When the player kills 3 of those so there is only 1 (Min COunt) left, the Wave Manager will spawn more until there are 4 (Mac Count) again. Once the Wave Manager has spawned 8 (Pool Size) robots total for this group, no more robots will be spawned for the group.



Result

Robots with pistols will now spawn and attack the player.

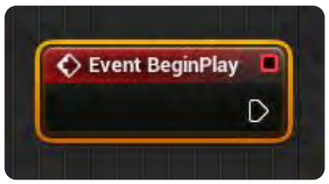
Previous Step
Adding Gameplay To Your Map
Next Step

4. Level Startup Scripting

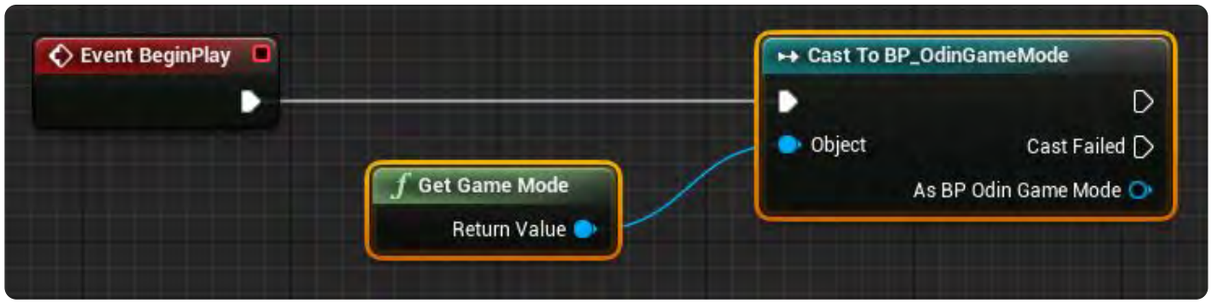
When the level begins, you want the player to start at the location of the PlayerStart Actor you placed previously. In order to make this happen, you need to move the player to that location and orient them correctly.

1. In the Level Editor toolbar click the **Blueprints** button and then choose **Open Level Blueprint** to open the script for the current level in the Blueprint Editor.

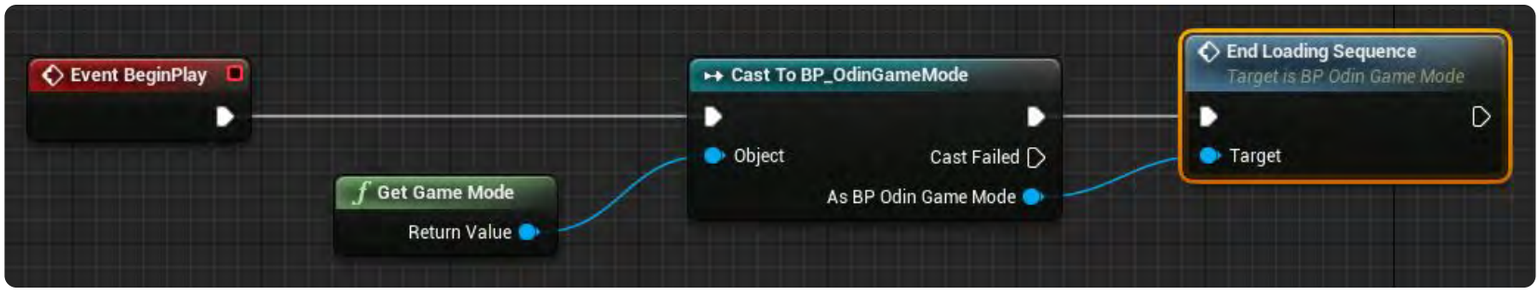
2. Right-click in the Event Graph and choose **Event BeginPlay** under **Add Event**.



3. Right-click in the Event Graph and choose **Get Game Mode** to get a reference to the current Game Mode instance. Drag off the **Return Value** pin and choose **Cast to BP_OdinGameMode**. Connect the exec output of the **BeginPlay** event to the exec input of the **Cast to BP_OdinGameMode** node.

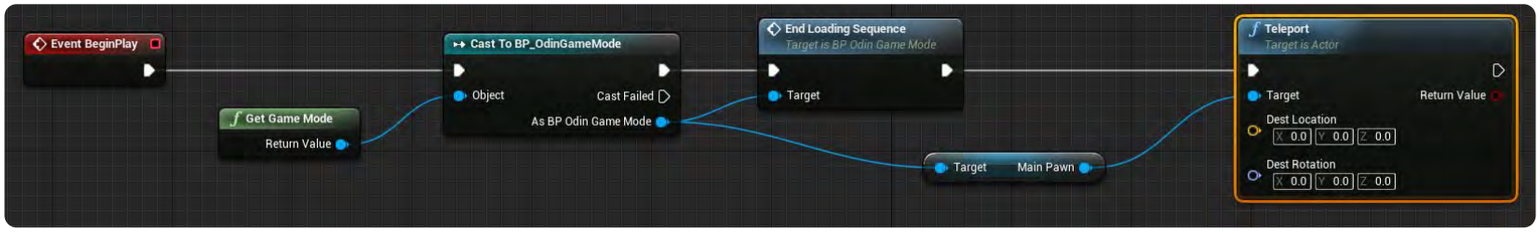


4. Drag off the **As BP Odin Game Mode** pin and choose **End Loading Sequence**. This performs some necessary cleanup after loading the level.

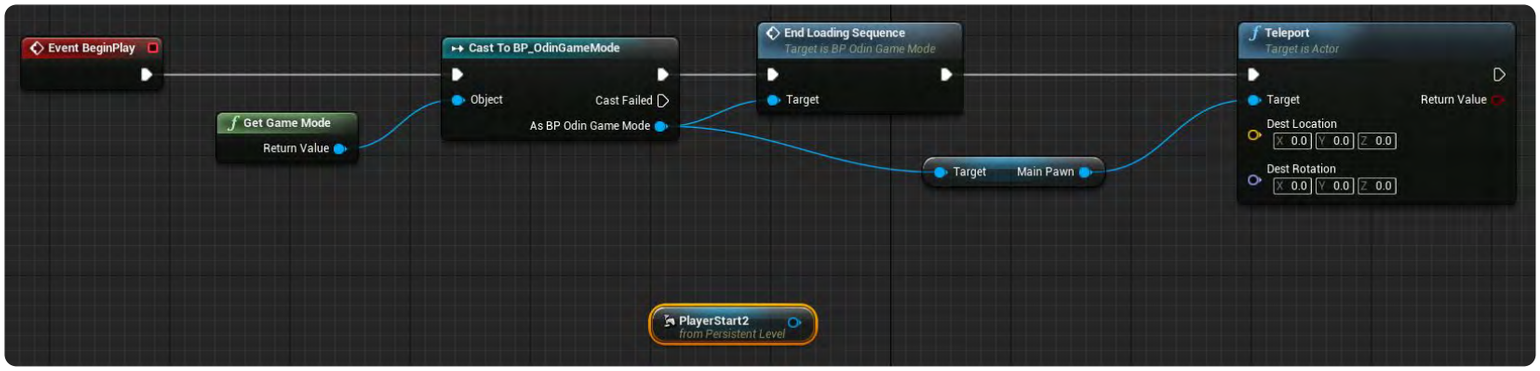


5. Drag off the **As BP Odin Game Mode** output and choose **Get Main Pawn** (under the **game** category). This gives you a reference to the player which you can use to set their location.

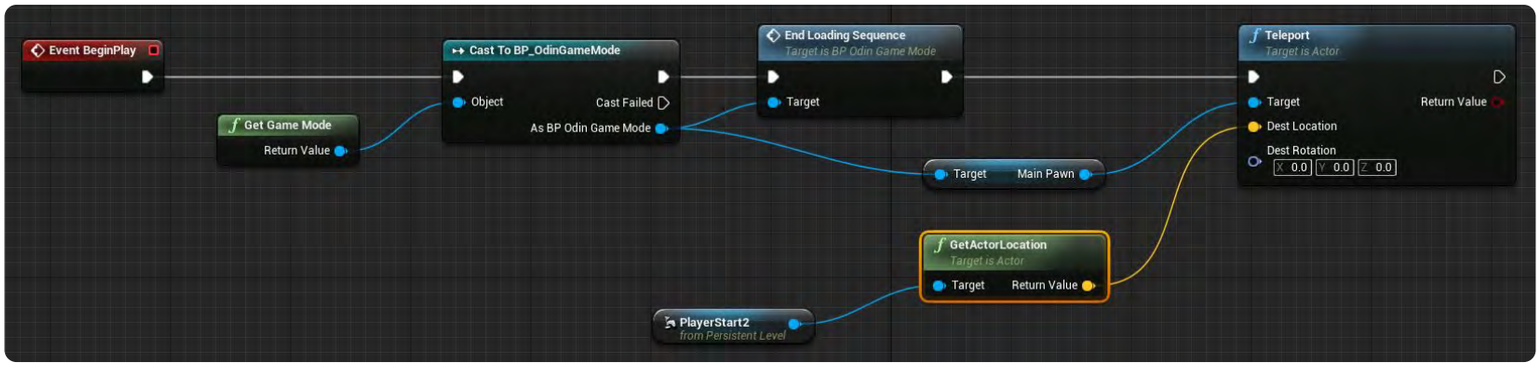
6. Drag off the **Main Pawn** pin and choose **Teleport** (under the **Utilities > Transformation** category). Connect the exec output of the **End Loading Sequence** node to the exec input of the **Teleport** node.



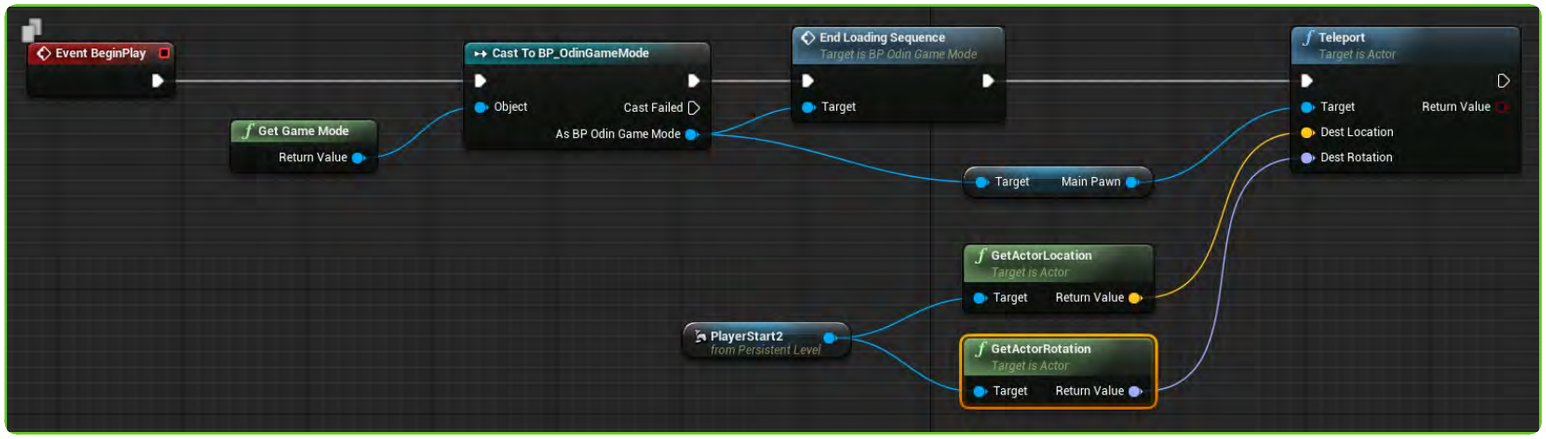
7. In the Level Editor Viewport, select the **Player Start** Actor you place previously. Back in the Blueprint Editor, right-click in the Event Graph and choose **Create a Reference to [ActorName]**.



8. Drag off the output pin on the Player Start reference and choose **Get ActorLocation**. Connect the **Return Value** output to the **Dest Location** input of the **Teleport** node.



9. Drag off the output pin on the Player Start reference and choose **Get ActorRotation**. Connect the **Return Value** output to the **Dest Rotation** input of the **Teleport** node.



Click the icon in the upper left corner of this image to copy the Blueprint Graph and paste it into your project.

Result

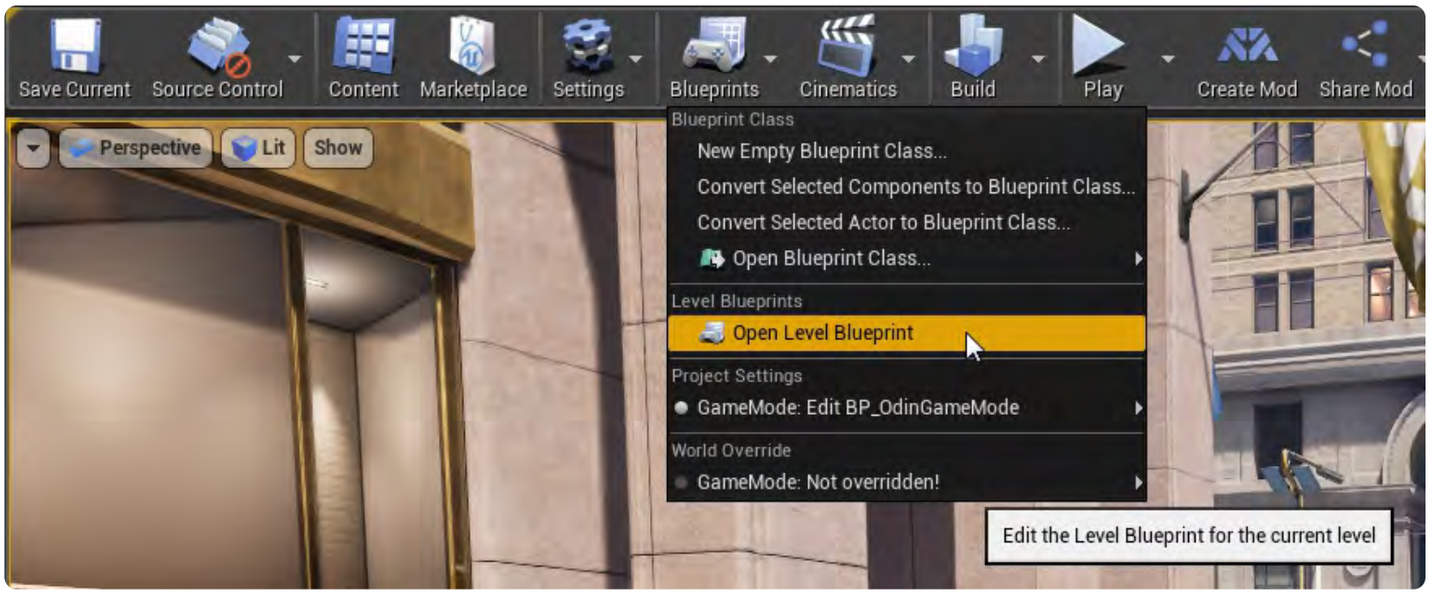
When you click **Play** in the Level Editor toolbar , you will now start off in the location of the Player Start Actor, facing the direction of the oncoming robots.

Previous Step | Adding Gameplay To Your Map | Next Step

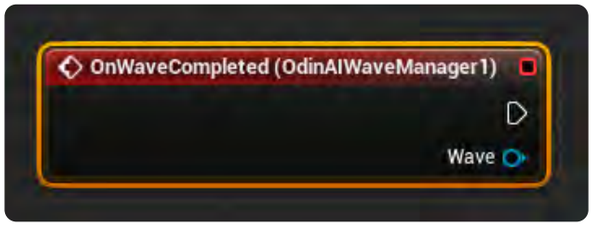
5. Level Completed Scripting

When all the robots in the wave have been recalled and the level is complete, you want the player to return to the Hub and continue the normal flow of the game.

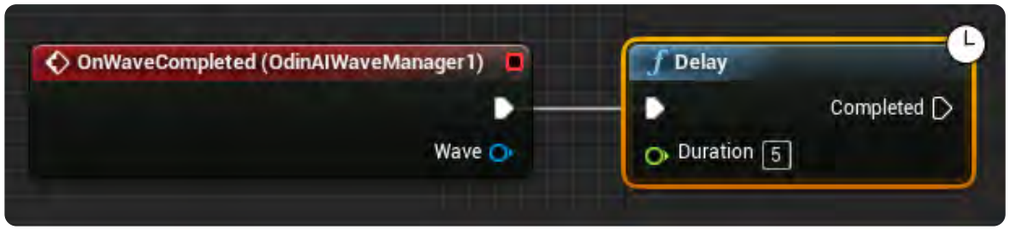
1. In the Level Editor toolbar click the **Blueprints** button and then choose **Open Level Blueprint** to open the script for the current level in the Blueprint Editor.



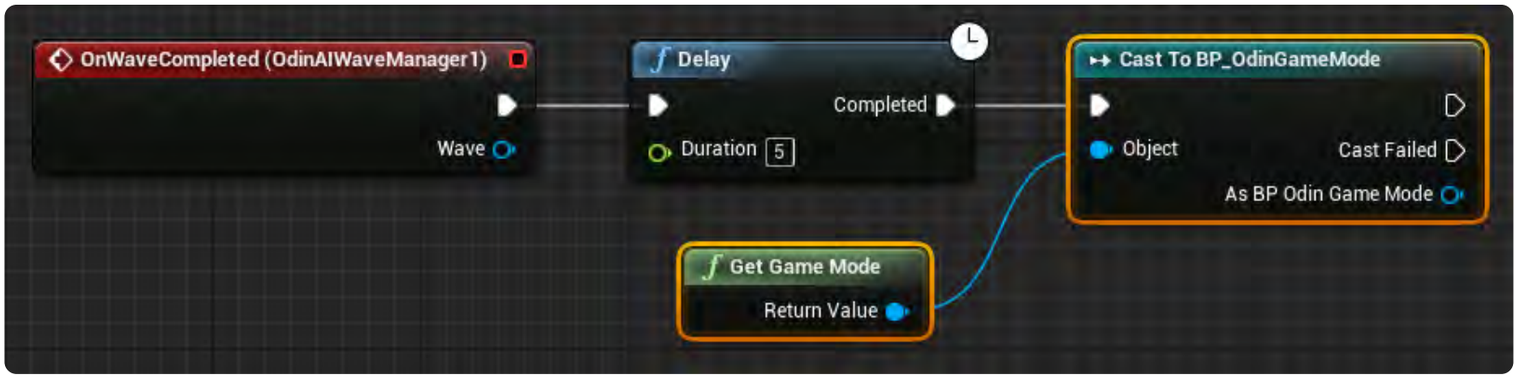
2. In the Level Editor Viewport, select the **Odin AI Wave Manager** Actor you place previously. Back in the Blueprint Editor, right-click in the Event Graph and choose **Add On Wave Completed**.



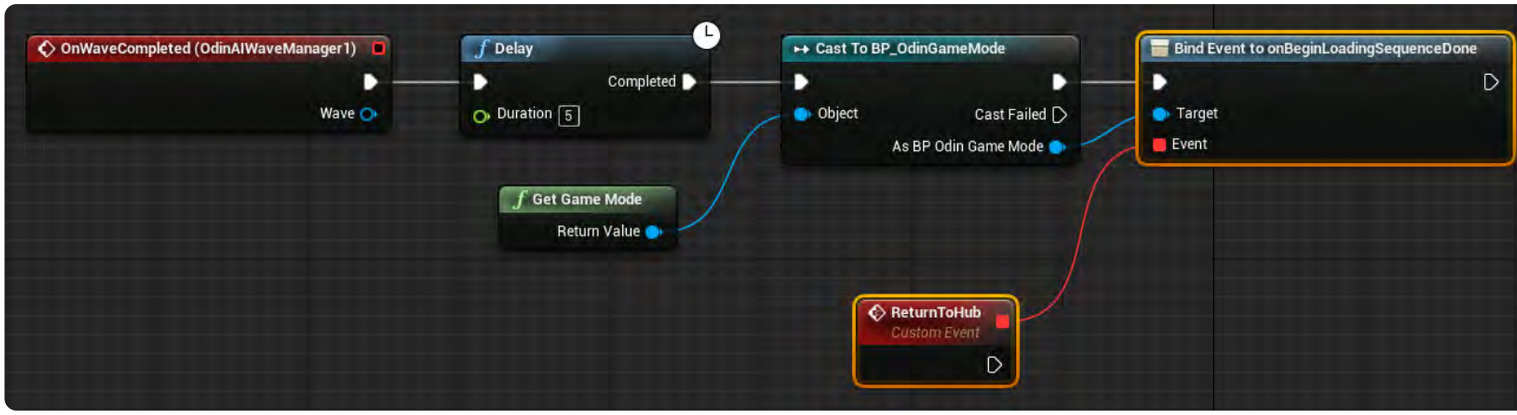
3. Drag off the exec output and choose **Delay**. Set the **Duration** to `5.0`. This will allow a brief period after the level is complete before the player is taken back to the Hub.



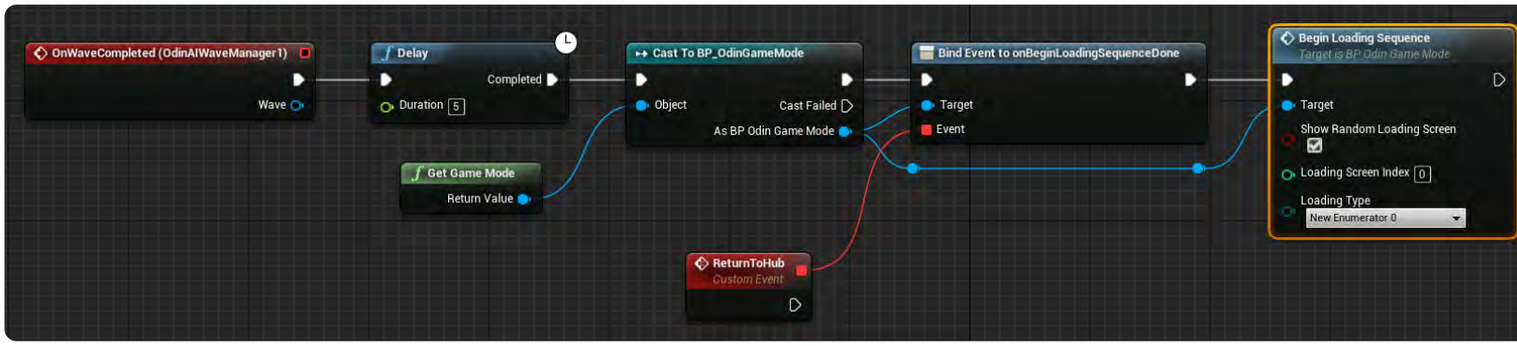
4. Right-click in the Event Graph and choose **Get Game Mode** to get a reference to the current Game Mode instance. Drag off the **Return Value** pin and choose **Cast to BP_OdinGameMode**. Connect the exec output of the **Delay** event to the exec input of the **Cast to BP_OdinGameMode** node.



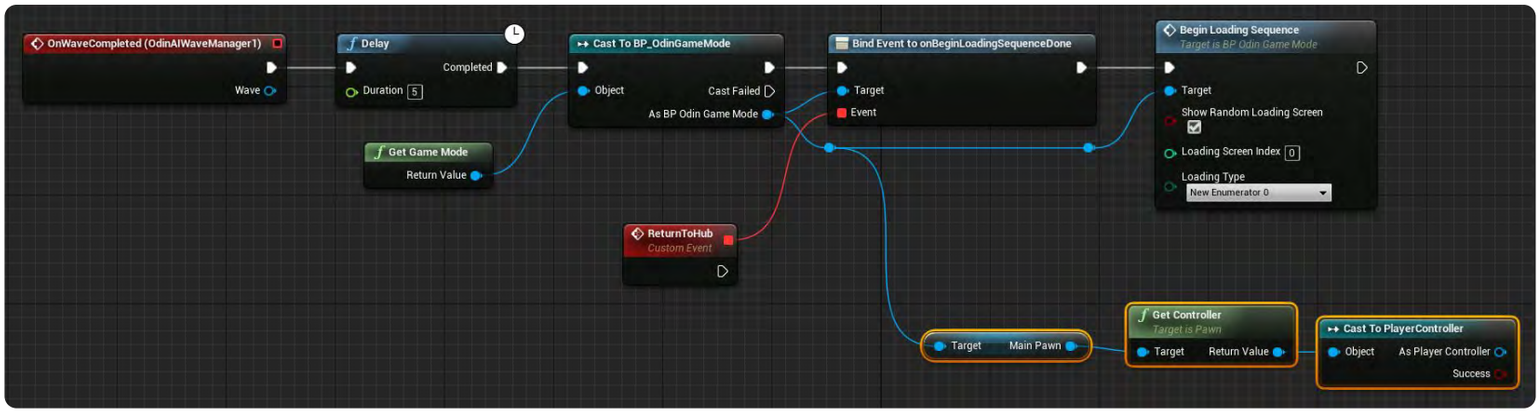
5. Drag off the **As BP Odin Game Mode** pin and choose **Bind Event to onBeginLoadingSequenceDone**. Drag off the **event** input and choose **Add Custom Event**. Name the event **ReturnToHub**.



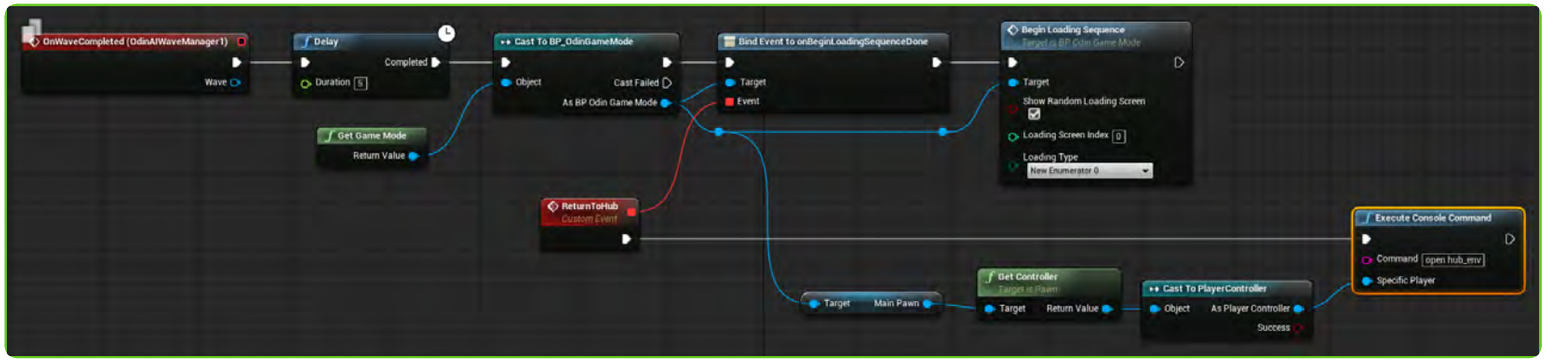
6. Drag off the exec output and choose **Begin Loading Sequence** to start the loading screen. Connectthe **Target** input to the **As BP Odin Game Mode** output from the cast node.



7. Drag off the **As BP Odin Game Mode** output on the cast node and choose **Get Main Pawn** to get a reference to the player. Drag off **Main Pawn** and choose **Get Controller** to get a reference to the player's Controller. Drag off **Return Value** and choose **Cast to Player Controller** to make sure it is the Controller of a human-controlled player, which is necessary to execute a console command.



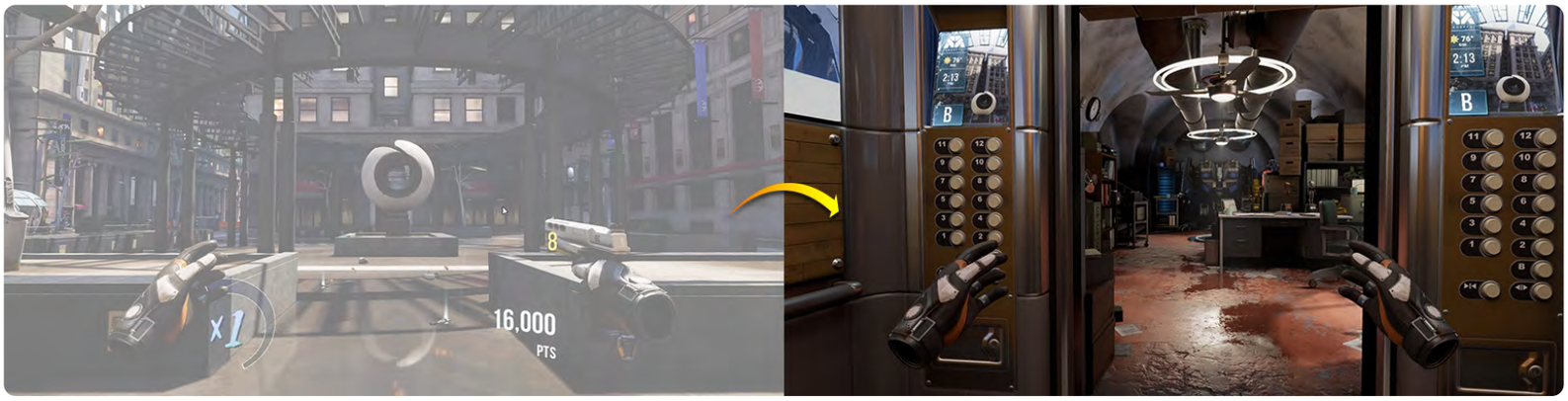
8. Drag off the exec output of the ReturnToHub event and choose **execute Console Command**. Enter *open hub_env* in the **Command** property. This will send a command to open the Hub level. Connect the **Specific Player** input to the **As Player Controller** output on the cast node.



Click the icon in the upper left corner of this image to copy the Blueprint Graph and paste it into your project.

Result

When the wave is complete, the loading sequence begins and the player is taken to the Hub.



Previous Step | Adding Gameplay To Your Map | Next Step

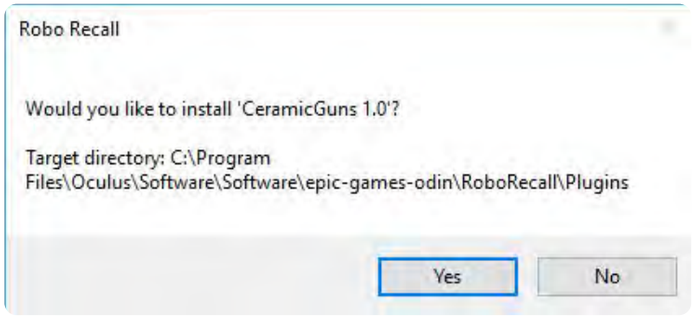
Installing a Robo Recall Mod

Mods you create yourself, or those you download from others, must be installed in order to be available in the game.

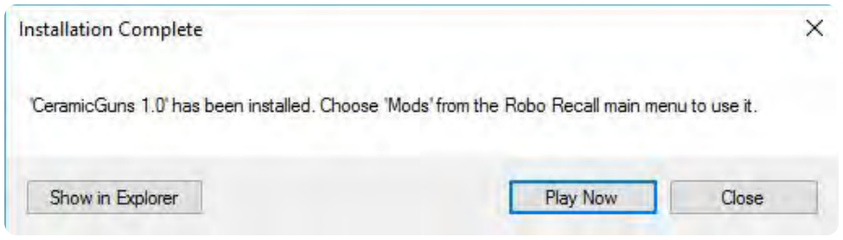
> The installation process works by associating the mod filetype with the game. You must play the game at least once for the association to be set up.

1. Locate the `.robo` file fopr the mod you want to install.

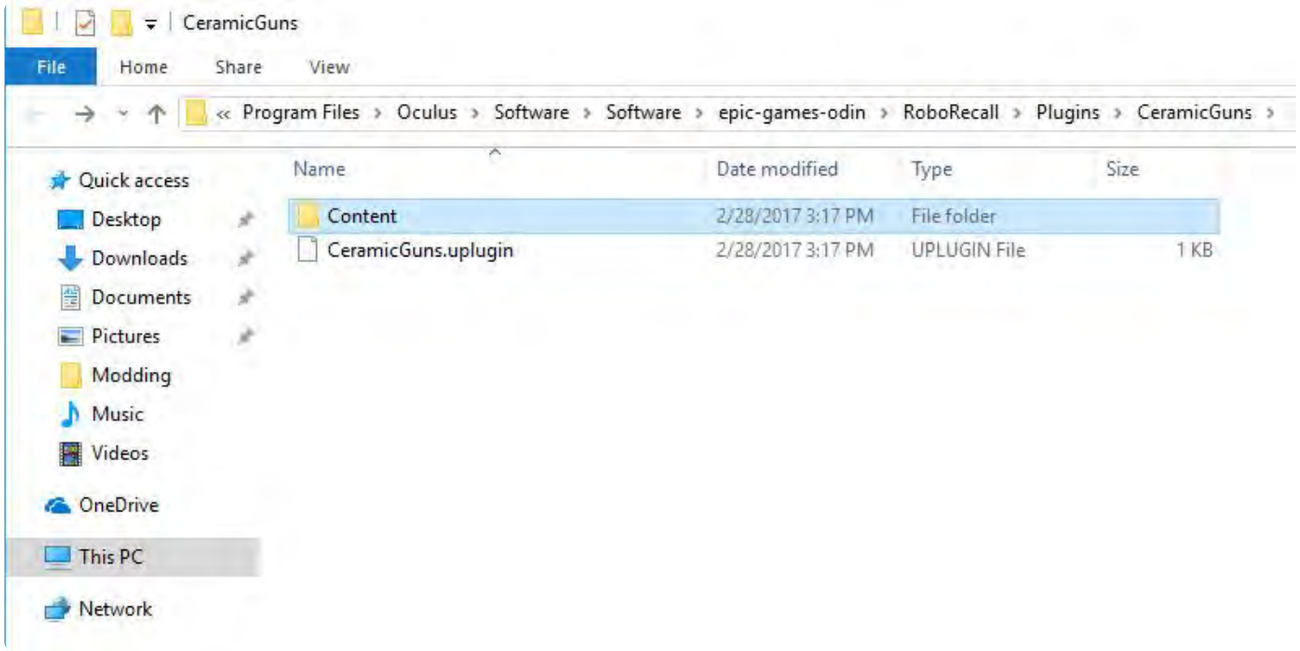
2. Double-click the `.robo` file and click **Yes** to begin the installation.



3. When the installtion is complete, you have the option to jump right into the game, view the install folder, or exit.



The install folder contains all of the files for the mod. You should see a `.uplugin` file along with a Content folder in the root directory.



4. The mod you installed is now available in the **MODS** menu in the game.

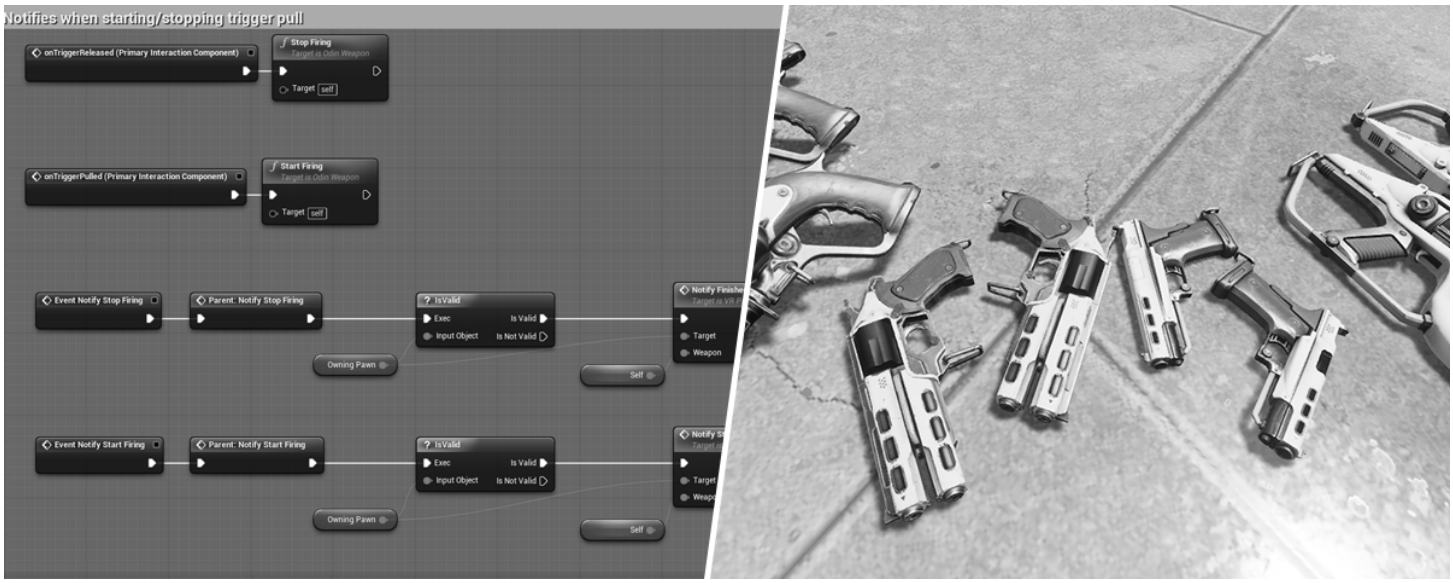
Troubleshooting Robo Recall Mods

On this page:

How To Uninstall a Mod That Crashes

1. Navigate to your Robo Recall install directory and open the **Plugins** folder.
2. Locate the folder for the mod that crashes and delete it.
3. Run the game and go to **MODS** in the Holostation menu. The mod should no longer appear in the list of installed mods.

Base Gun Reference



Events

Event Notify Start Firing

The Notify Start Firing event is called each time the player presses the 'fire' button. Any nodes connected to the *exec* output pin will be executed at this time. Use this event to extend or override the firing functionality of the weapon, such as changing the firing location or spawning firing effects.

Outputs

Type	Description
exec	Nodes connected to this are exected when the event is called.

Event Notify Stop Firing

The Notify Stop Firing event is called each time the player releases the 'fire' button. Any nodes connected to the *exec* output pin will be executed at this time. Use this event to extend or override the 'stop firing' functionality of the weapon, such as cleaning up firing effects spawned when firing.

Outputs

Type	Description
exec	Nodes connected to this are exected when the event is called.